STOCHASTIC MODEL BUILDING OF PHOTOPHYSICS USING HIDDEN MARKOV MODEL

A Thesis
Presented to
The Academic Faculty

by

Soonkyo Jung

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of School of Chemistry and Biochemistry

Georgia Institute of Technology
August 2015

STOCHASTIC MODEL BUILDING OF PHOTOPHYSICS USING HIDDEN MARKOV MODEL

Approved by:

Dr. Robert M. Dickson, Advisor
School of Chemistry and Biochemistry
Georgia Institute of Technology

Dr. Mohan Srinivasarao
School of Materials Science and Engineering
Georgia Institute of Technology

Dr. Thomas Orlando
School of Chemistry and Biochemistry
Georgia Institute of Technology

Dr. Harold Kim
School of Physics
Georgia Institute of Technology

Dr. Joseph W. Perry
School of Chemistry and Biochemistry
Georgia Institute of Technology

Date Approved: May 5, 2015

ACKNOWLEDGEMENTS

I am grateful to my advisor Dr. Robert M. Dickson for giving me an opportunity for research in an amazing area, and motivating, guiding, pushing, and supporting me. I also would like to appreciate my committee members, Dr. Joseph Perry, Dr. Thomas Orlando, Dr. Mohan Srinivasarao, and Dr. Harold Kim for their helpful advice and useful comments.

I'd like to appreciate the past and current Dickson group members for their help and kindness, Dr. Junhua Yu, Dr. Chris Richards, Dr. Sungmoon Choi, Dr. Rusty Nicovich, Dr. Amy Jablonski, Dr. Saugata Sarkar, Yen-Cheng Chen, Aida Demissie, Blake Fleischer, Jung-Cheng Hsiang, Tzu-Hsueh Huang, Daniel Mahoney, and Joe Richardson. I hope everything's going well for them.

I would like to thank Dr. Seung Soon Jang and all my friends in Georgia Tech for their kindness and encouraging me. I also appreciate Cue Jung, Honggab Kim and my friends in Korea.

Finally, I would like to thank my family for their endless support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF SYMBOLS

A	Transition matrix
a_{ij}	Transition probability from state *i* to state *j*
$\bar{a}_{ij}$	Trained transition probability from state *i* to state *j*
$\alpha_t(i)$	Forward probability in state *i* at time *t*
B	Emission matrix
$b_i(x)$	Emission probability of observing *x* at state *i*
$\bar{b}_i(x)$	Trained emission probability of observing *x* at state *i*
$\beta_t(i)$	Backward probability in state *i* at time *t*
$C_{ij}(\tau)$	Correlation coefficient of species *i* and *j* at lag time τ
$Dark_n$	*n*-th dark state
$\delta_t(j)$	The probability of $\psi_t(i)$
Φ_{Dark1}	Quantum yield of $Dark_1$
Φ_{Det}	Detection efficiency
Φ_{FL}	Fluorescence quantum yield
k_{off}	Escape rate from off-state
k_{on}	Escape rate from on-state
λ	model parameters of HMM
O	Observed data
o_t	Observable at time *t*
π	Initiation matrix
q_t	the hidden state at time *t*
Q	Hidden state sequence

q_t	Hidden state at time *t*
S_0	Ground singlet state
S_1	The lowest, excited singlet state
τ_{off1}	Lifetime of $Dark_1$
τ_{off}	Off-time
τ_{on}	On-time
$\psi_t(i)$	The most probable state path that ends in state *i* until time *t*

LIST OF ABBREVIATIONS

AIC	Akaike information criteria
APD	Avalanche photodiode
BIC	Bayesian information criteria
BW	Baum-Welch algorithm
DNA	Deoxyribonucleic acid
EM	Expectation-maximization
FCS	Fluorescence correlation spectroscopy
FRET	Förster resonance energy transfer
GFP	Green fluorescent protein
HMM	Hidden Markov Model
HQC	Hannan-Quinn information criteria
LE	Localization error
MMPP	Markov-modulated Poisson process
PbPHMM	Photon-by-photon hidden Markov model
PBL	Poisson-modificated BW algorithm with BIC and LE
PB	Poisson-modificated BW algorithm with BIC
PC	Poisson-modificated BW algorithm with χ^2 probability
PCL	Poisson-modificated BW algorithm with χ^2 probability and LE
PCR	Polymerase chain reaction
PET	Positron emission tomography
PDF	Probability density function
PMF	Probability mass function

PMT	Photomultiplier tube
TCSPC	Time-correlated single photon counting

SUMMARY

Exogenous fluorescence probes have been widely employed in studying complex biological dynamics with high temporal and spatial resolution. One of the limiting factors is a photoinduced dark state, in which dynamic information is lost. The dark state, however, can be useful if it can be utilized for a particular purpose through optical modulation. Transferring this concept to more traditional fluorophores requires a detailed understanding of the photophysical dynamics. Because photon emission is a random process corresponding to underlying photophysical state transitions, a robust, probabilistic method is required to uncover the most probable model. This thesis focuses on a generalized hidden Markov model (HMM)-based method to build a proper photophysical model, and calculate model parameters from simulated and experimental data.

The conventional algorithm in HMMs is modified to analyze Poisson-distributed intensity trajectories. Several types of criteria were used to determine the "true" number of states and compare to known values of simulated data sets. By modifying the Baum-Welch algorithm to introduce chi-square probability and localization error, we have improved HMM performance both in determining the dimensions of unknown systems and in robustness even if the intensity trajectory is very short, or if poor initial conditions are used.

Experimental conditions including non-unity detection efficiency and background noise lead to non-Poisson distribution of intensity counts, which cannot be explained by Poisson statistics. Photon-by-photon HMM (PbPHMM) was developed to build a

photophysical model from non-Poisson distributed photon time traces, with the goal of incorporating the information often discarded by time-binning. Additionally, in PbPHMM, the solution of the master equation is not required to extract the rates of underlying dynamics, which is often long and complicated. The fitting ability of PbPHMM was evaluated by analyzing simulated time traces of fluorescence photons, varying experimental and photophysical parameters. The relation between photophysical parameters and the trained model parameters were formulated based on probabilistic theory. The number of dark states was determined by Bayesian information criteria. When multiple time traces are available, the fitting performance of PbPHMM could be improved, showing relative error of parameter estimation less than 10%.

PbPHMM was applied to build a photophysical model building from photon time traces from optically-modulated silver nanodots and excited AcGFP. One bright and one dark state were predicted from photon trajectories from silver nanodots. The information criteria confirmed that two dark states at different time scales existed in a photophysical model for AcGFP.

CHAPTER 1

INTRODUCTION

1.1 Motivation: fluorescence imaging

Optical imaging has been widely applied to study complex and heterogeneous biological dynamics [1-7]. Among various optical imaging methods, fluorescence imaging has become a popular method because signals from multiple fluorophores at different wavelengths can easily be separated. Since human genes, separated by ~1Mb, were visualized during hybridization [8], fluorescence imaging has been employed in detecting tumors with near-IR copolymer probes [9], studying giant unilamellar vesicles from preferentially labeled fluid phase and estimating boundary tension [10], *in vivo* imaging with semiconductor quantum dots [11, 12], intravital microscopy [6, 13-15], and superresolution microscopy [16-21]. Diverse kinds of fluorophores with a broad range of wavelengths have been developed including organic dyes [22, 23], semiconductor quantum dots [24, 25], silver nanodots [26, 27], and fluorescent proteins [28, 29].

One of the challenges that limit the amount of information from fluorescence imaging is a photoinduced dark state. In the dark state, fluorophores do not emit photons and therefore become at least temporarily invisible. Several mechanisms including intersystem crossing to triplet states [30, 31] and nonradiative Auger process [32, 33] have been proposed to describe dark state dynamics. Even in particle tracking, dark state residence prevents continuous tracking of probe trajectories. Such temporary disappearances of fluorescent probes may result from exiting out the field of view,

blinking, or photobleaching, but without quantitative understanding and characterization of these non-emissive states, interpretation of behavior remains clouded.

The dark state, however, can be useful if it can be adjusted for a particular purpose. Recently, the Dickson lab at the Georgia Institute of Technology has demonstrated that we can control dark state lifetimes through optical modulation to selectively enhance fluorescence signals of interest [34, 35]. In this scheme, a high energy laser excites a molecule to generate lower energy fluorescence, while inadvertently building up dark state population. This dark state will naturally decay on its own to regenerate the original ground state. At high primary excitation intensities, however, the rate of dark state generation overpowers the rate of dark state decay, and a large dark state population maintains a steady state. Optical modulation results from co-illumination with a very low energy secondary laser that depopulates the dark state faster than it would naturally decay, thereby enhancing overall fluorescence intensity through ground state repopulation. The dark state depopulation rate is proportional to the intensity of a secondary laser with lower energy than primary emission, and the optical modulation enables to recover fluorescence signals effectively [34]. Since the secondary laser is lower in energy than that of the collected fluorescence, no additional background is generated. Currently, only a small subset of fluorophores yield significant improvements through optical modulation. Thus, transferring this concept to more traditional fluorophores requires a detailed understanding of photophysical dynamics.

Although correlation analysis is commonly used to extract photophysical parameters and build models, proper interpretation relies on the assumption of a specific underlying photophysical model. Without assumptions of the underlying state

connectivity and even the number of accessible states, a robust, statistical method is required to uncover the most probable model. In this thesis, I propose a generalized hidden Markov model (HMM)-based method to build a proper model of optical modulation, and calculate model parameters based on simulated and experimental data.

1.2 Photophysical processes of fluorescent molecules

Fluorescence is one of the photophysical processes in which an atom or a molecule emits a photon that results from excitation by absorbing a photon retaining spin multiplicity [36]. The processes can be visualized by the Jablonski diagram in Figure 1.1. A molecule at ground singlet state S_0 absorbs a photon and is electronically excited to one of the vibrational levels in excited singlet state S_1 while the spin multiplicity is maintained due to the selection rule for electronic transition. The excess energy in the molecule at S_1 is redistributed to its environments, and the molecule steps down to the lowest vibrational level of S_1. From this excited state, two competitive pathways exist to return to the ground state S_0: a radiative and a nonradiative pathway. The radiative de-excitation pathway yields fluorescence in which the molecule emits a photon with an energy that corresponds to the energy difference between S_0 and S_1. Due to nonradiative vibrational relaxation to the lowest vibrational level of S_1 after excitation, the energy of emitted photon is lower than that of the absorbed photon. Therefore the emission spectrum is red-shifted relative to the absorption spectrum (Stokes shift). The typical time scale of excited state lifetime is 10^{-8}~10^{-10} s, and such short lifetime provides the fast time resolution to fluorescence-based techniques [36, 37].

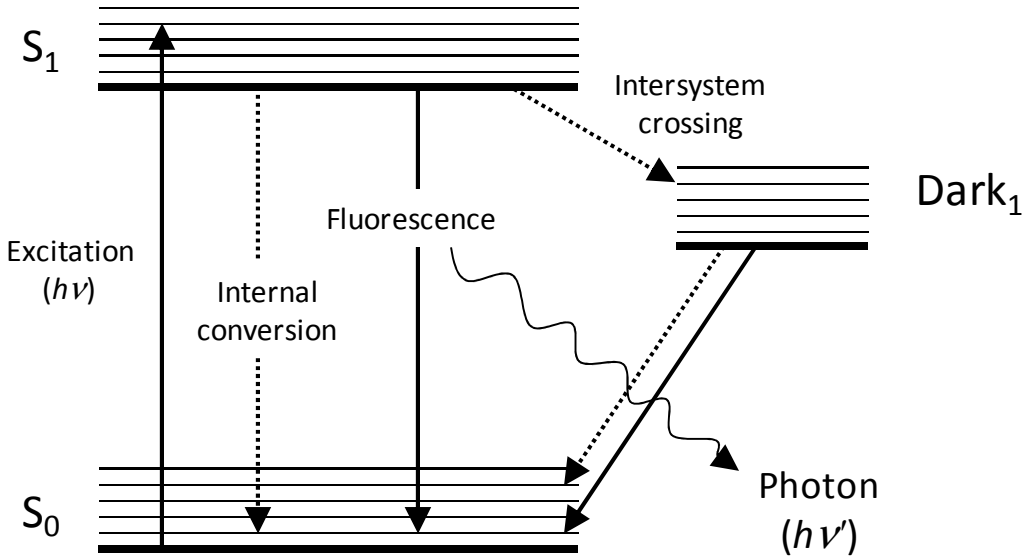

Figure 1.3 Jablonski diagram for photophysical processes in fluorescent molecules.

Another competitive pathway to the ground state is internal conversion. Internal conversion is a nonradiative transition between states with same multiplicity and its lifetime of transition from S_1 to S_0 is typically 10^{-11}~10^{-6} s [38]. A fluorescence quantum yield Φ_{FL} is defined by the ratio of rates of competing radiative and nonradiative pathways. Φ_{FL} also can be calculated by the ratio of the number of emitted photons to the number of absorbed photons [36].

Diverse kinds of fluorophores including organic dyes, quantum dots, fluorescent proteins [28, 29], silver nanodots [26, 27], and nitrogen-vacancy centers in nanodiamonds [39] have been developed for fluorescence imaging. The ideal fluorophores need to have (1) a large extinction coefficient for efficient excitation at a given wavelength, (2) a large fluorescence quantum yield, and (3) sufficient photostability. An absorption cross section σ (with units of cm^2) is the quantum interpretation of the molar extinction coefficient ε (with units of M^{-1} cm^{-1}), which indicates the probability of a photon being absorbed by molecules in a given area (cm^2) [40]. Another requirement for fluorophores is a large fluorescence quantum yield, Φ_{FL}. Typical values of ensemble fluorescence quantum yield range from 0.04~0.4 for cyanine dyes [38], 0.13~0.40 for silver nanodots [41-43], 0.02~0.8 for fluorescent proteins [44], 0.2~0.7 for quantum dots, and near unity for xanthene derivatives [37]. Moreover, fluorophores need to be photostable enough for sufficient photon collection before it irreversibly converts into non-emissive state via a photochemical process called photobleaching. The number of excitation/de-excitation cycle of organic dyes prior to photobleaching ranges from 2×10^3 for coumarins to 10^5 for Cy5 to 3×10^6 for rhodamine dyes [22, 45] and ~10^6 for GFP [46]. In most cases, photobleaching is induced by the interaction of fluorophores with reactive oxygen species,

which are generated from the reaction of fluorophores at triplet states with molecular oxygen [22, 37, 47]. Although transition from ground triplet state to excited singlet state is forbidden for molecular oxygen, energy transfer between ground state molecular oxygen and other species in their triplet states makes it easier to generate excited singlet oxygen molecules [22]. The photostability of fluorophores can be enhanced by using enzymatic oxygen scavengers and triplet quenchers [48-50] or protective agents tethered to fluorophores molecules [37, 51, 52].

1.3 Photoinduced dark state

Although bright states yield strong fluorescence upon excitation, not all decay processes proceed within the bright state manifold. Photoinduced dark states are also accessible, often in low yield, from the excited level and can result in steady-state dark states with significant population. These dark states may limit bulk fluorescence rates. Unless large dark state populations are achievable, dark states remain largely undetectable as the net absorption often remains too low to be seen with transient absorption spectroscopy. Consequently, single molecule experiments are now widely utilized to characterize such photoinduced dark states, which have been invoked in explaining, for example, intensity fluctuations on the single molecule level [53, 54]. Since the first single molecule studies characterizing such discrete intensity fluctuations from single pentacene molecules in para-terphenyl crystal at room temperature [30] and a single terrylene molecule at cryogenic temperature [55], studies have been performed to explain the origins of photoinduced dark states in a wide range of systems [56-61].

A well-known explanation of dark states is the intersystem crossing to triplet levels. The molecule in a triplet state does not fluoresce until relaxation to the ground singlet

state and subsequent excitation-relaxation within the singlet manifold. The lifetime of these triplet states ranges from microseconds to milliseconds, as exemplified in organic dyes such as Rhodamine 6G [56], carbocyanine [58], and DiI molecules [31]. Even with the small intersystem crossing quantum yield, significant fractional populations can be trapped in triplet dark state, limiting the fluorescence intensity [37]. Based on a two-state model with a bright and dark state, the dark state lifetime can be determined by fluorescence autocorrelation studies [62-64].

Another type of fluorescence fluctuation has been studied since fluorescence intermittency has been observed from single colloidal quantum dots [65]. Unlike molecular states, quantum dot photoluminescence signals appeared to be similarly random telegraph signals, but with widely distributed (~ms to hrs) time scales [32]. The proposed mechanism to describe the long-lived dark state is an ionization-recombination model [33, 66-68]. When a single ground state quantum dot absorbs a photon, an electron-hole pair is generated – Auger ionization. This long-lived ionized state is followed by Auger recombination. During this process, the hole is trapped in one of the active recombination centers on the quantum dot surface, and the photoexcited electron-hole pair recombines in a nonradiative way [32, 33]. Then the recombination is followed by the relaxation to ground state. The time scale of nonradiative recombination is several picoseconds [32]. The energy gap between delocalized electronic levels causes hole trapping rates to fluctuate [67]. The lifetime of this kind of dark state has been observed to follow inverse power law distribution [66, 69].

In a bulk experiment, transitions into dark states should be synchronized to measure dark state lifetime. However, initiating a dark state transition is an uncontrollable

stochastic process that is governed by dark state quantum yield. To circumvent the need for synchronization of uncorrelated stochastic processes in bulk experiments, single molecule experiment enable measurement of molecular parameters such as dark state quantum yield or lifetime. Since its first measurement more than twenty years ago [70, 71], single molecule spectroscopy has been extensively applied to study complex dynamics at the molecular scale. Its applications were reported on measuring of protein binding and rupture force using force spectroscopy [72-75], observing conformation change in helicase protein and DNA junction using Förster resonance energy transfer (FRET) [76, 77], and gating kinetics of a single ion channel [78]. Collecting signals from a single molecule allows one to avoid ensemble averaging. This field has been enhanced by the development of low noise amplifiers and sensitive and fast detectors, such as photomultiplier tubes (PMTs), avalanche photodiodes (APDs), and electronic amplifiers [79], which can convert individual photons or electrons into signal pulses.

Typical signals from a single molecule, however, are weak and short. Due to photobleaching, an insufficient number of data points are often collected in fluorescence microscopy. In order to increase photostability of fluorophores, cyclooctatetraene was used as a triplet state quencher to reduce the lifetime of triplet dark state [49]. Additionally, since molecular oxygen causes photobleaching and photooxidation for organic dyes, enzymatic oxygen scavengers with β-mercaptoethanol or Trolox were introduced to suppress blinking [48, 49].

1.4 Stochastic modeling

Since the first efforts to stochastically interpret burst signals of current across single ion channels [80], various stochastic methods have been applied to extract

parameters and build a model from noisy data. Model parameters can be extracted from raw data [11, 31, 43, 56, 63, 81-99] or idealized data [100-109], using fluorescence correlation spectroscopy (FCS) [11, 31, 43, 56, 63, 81-95], linear and nonlinear filtering [100-104, 109], intensity change point detection [105, 107, 108], and Markov-modulated Poisson processes [110-115].

1.4.1 Raw data analysis

FCS is a technique in which underlying processes are characterized from the correlation function of fluorescence fluctuations [81]. In 1970s, FCS was applied to estimate rate constants of dye-DNA complex formation from the autocorrelation function of fluorescence fluctuation [82]. Since the first application successfully revealed kinetic information with sufficient precision, FCS has been widely utilized in different areas including monitoring enzymatic dynamics [63, 83, 84], determining the number of fluorescent particles and size distribution of quantum dots [11], and monitoring dynamics of the photophysical and photochemical fluctuations in organic dyes [31, 56, 85, 86], GFP [87-89], and silver nanodots [43, 90]. Combined with stimulated emission depletion (STED) nanoscopy, FCS was used in observing single lipid molecules diffusing in the plasma membrane within diffraction limited area [91].

Time-correlation function analysis has conventionally been used to extract temporal information about the characteristic timescales of a given system. The *n*th order correlation function $C_{X^1X^2\cdots X^n}(t_1, t_2, \cdots, t_n)$ of stochastic variables $X^1(t_1)$, $X^2(t_2)$, $\cdots, X^n(t_n)$ at time $t_1, t_2, \cdots, t_n$ is defined by:

$$C_{X^1X^2\cdots X^n}(t_1, t_2, \cdots, t_n) = \langle X^1(t_1)X^2(t_2)\cdots, X^n(t_n)\rangle \quad (1.1)$$

where $\langle ... \rangle$ means averaging over time. Within a given photophysical model, a second-order correlation function can be used to characterize the temporal fluctuations or the relation of signals from two species [31, 63, 82, 87, 92-94]. The correlation function, $C_{ij}(\tau)$, of the fluorescence intensity signal from species i at time t, $I_i(t)$, and species j with lag time τ, $I_j(t+\tau)$, is given by:

$$C_{ij}(\tau) = \frac{\langle I_i(t) I_j(t+\tau) \rangle}{\langle I_i(t) \rangle \langle I_j(t) \rangle} = \frac{\langle I_i(0) I_j(\tau) \rangle}{\langle I_i(0) \rangle \langle I_j(0) \rangle} \tag{1.2}$$

The second identity is obtained when the random process is stationary. When $i = j$, eq. (1.2) is called an autocorrelation function.

Fluorescence intensity fluctuations also depend on the fluorophore photophysical dynamics for all times shorter than it remains in the focal volume. However, due to the difference in time scale, the autocorrelation formula can be divided into two parts: diffusion and photophysical processes. The diffusive part of autocorrelation formula does not appear for immobilized fluorescent particles. The fluorescence intensity is proportional to the number of fluorescent particles or the probability of photon detection in case of single molecule experiment. Thus, the autocorrelation function of fluorescence intensity can be derived from the master equation for photophysical processes (see Appendix A) [116, 117]. The solution of master equation for on-off dynamics is the conditional probability of photophysical state at time $t + \tau$ given the state at time t [93].

With the assumption of stationary process, the intensity autocorrelation function of the following reaction is given by eq. (1.3) where τ_{on} and τ_{off} are the inverse of escape rate k_{on} and k_{off} from on and off state, respectively [31, 95]. Solving this set of coupled

rate equations is straightforward for the simple case of a single bright and a single dark state:

$$\text{on} \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} \text{off} \qquad G(\tau) = A + Be^{-\tau/t_c}$$

$$\frac{1}{t_c} = \frac{1}{\tau_{\text{on}}} + \frac{1}{\tau_{\text{off}}}, \quad \frac{A}{B} = \frac{\tau_{\text{on}}}{\tau_{\text{off}}} \tag{1.3}$$

However, the problem in correlation analysis is that the solution of the master equation becomes extremely complicated if more than two photophysical states exist. In general, a model must be assumed and the correlation function fitted to this expected model to extract photophysical parameters, but the underlying model cannot be confirmed through correlation analysis. Thus, analysis is available only if the underlying model is known. Photophysical state transitions are not visible, and the only observables available are the events of detecting photons emitted upon singlet relaxation. Detected photons are related to and dependent upon the populations of bright and dark states. Therefore, while useful in cases with simple and known photophysics, autocorrelation methods are not generally applicable to fully understanding and characterizing photon trajectories.

Time traces of FRET efficiency have been theoretically studied by deriving distributions of the number of photons emitted from donor-acceptor pairs and energy transfer efficiency [96-99]. Various sources that cause FRET efficiency fluctuation include intersystem crossing to dark states, shot noise, conformation change, and translational diffusion [97]. FRET efficiency trajectories were decoded based on two conditions: (1) the independent FRET efficiency on the position of a molecule in laser spot and (2) the constant sum of counting rate of photons from donor and acceptor

channels. The joint probability distribution of donor and acceptor photons in a given bin could be approximately derived, which was converted into the function of FRET efficiency [97]. By obtaining energy transfer efficiencies and interconversion rates from maximum likelihood estimators, the theory was extended to explain two- and three-colored photon trajectories and estimate the rates of conformational change [98, 99].

1.4.2 Idealized data analysis

Rather than analyzing raw data, model parameters including transition probabilities and conversion rates can be more easily extracted from idealized data segmentations combined with threshold techniques or histogram fitting [109]. During these idealizing procedures, true trajectories are probabilistically revealed from noisy data. Different types of filters have been used to suppress noise, enhance signal-to-noise ratio, and expose hidden dynamics. DNA looping-unlooping kinetics were studied by observing the motion of tethered particles, revealing the interaction between DNA and wild-type Lac repressor [118]. Mechanical noise including stage drift or apparatus fluctuations can be removed by using low-pass filter with low cutoff frequency. Shot noise in measuring donor-acceptor distances in the yeast transcription factor was minimized by applying optimal filter in frequency space [101]. Low-pass filters can also be applied to increase signal-to-noise ratio of ion channel current recorded by patch clamp method [119, 120]. It should be noted that short-lived transitions are susceptible to omission when filters are applied heavily, which results in increased model parameters. To overcome the weakness of low-pass filtering, nonlinear forward-backward filtering was developed for reducing background noise in analyzing single-channel currents [102]. A nonlinear filter has been applied to reduce noise in FRET signals from conformational fluctuations of protein

molecules while keeping fast transitions [103, 104], and the filtered signals showed clearly-separated states in the histograms of FRET efficiency.

Maximum-likelihood estimation has been applied to analyze photon emission data by determining intensity change points [105]. Expectation-maximization (EM) is utilized to identify the change points. In this method two hypotheses are tested within the time window: the null hypothesis that states no change point exists, and the alternative hypothesis that states at least one change point exists. Following the hypothesis test, intensity levels are determined by maximum-likelihood estimation. After the intensity levels are optimized, the number of states has been determined by information criteria. This method has been applied to various fields [106-108], and has served as a useful tool to set a starting point to determine and reconstruct state dynamics. However, in real cases, photon time traces have non-Poisson characteristics due to imperfect experimental conditions, including non-uniform photon collection efficiency and background noise.

1.5 Hidden Markov models

1.5.1 Background

Molecular states cannot be observed directly; we can measure photon counts, intensity, or arrival times. A robust and unbiased method is required to reveal the hidden dynamics from experimentally collected photons. A hidden Markov model (HMM) is better suited to this task than are many other statistical approaches as the HMM model is mathematically simple to use and the HMM likelihood of observed data can be calculated in a short running time.

HMM is a doubly stochastic process in which a sequence of observations is determined by underlying hidden states [121-123]. A general HMM process is displayed

in Figure 1.2. The term "hidden" means that states are not directly observable. Only the sequence of observations $O=o_1, o_2, \ldots, o_{T-1}, o_T$ that represent hidden state sequence $Q=q_1, q_2, \ldots, q_{T-1}, q_T$ can be detected. Let N denote the number of hidden states, then the state at time t, q_t, is one of the N possible states s_i, $1 \leq i \leq N$. The hidden state sequence follows Markov property; that is, a current state q_t is only dependent on the previous state q_{t-1}.

$$P(q_t \mid q_1, q_2, \cdots, q_{t-2}, q_{t-1}) = P(q_t \mid q_{t-1}) \tag{1.4}$$

Originally developed for speech recognition [121], HMM enables reconstruction of the state sequence from observables when the true states are hidden. Iterative state reconstructions refine the three parameters –transition matrix A, emission matrix B, initiation matrices π– needed to describe time series with well-defined transition probabilities linking all states The transition matrix element a_{ij} is the probability that one state i changes to a different state j in the subsequent step.

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i) \tag{1.5}$$

The emission matrix B links the observable o_t at time t to hidden states s_i that, in fluorescence intensity trajectories, corresponds to the different observed intensity levels.

$$b_i(x) = P(o_t = x | q_t = s_t) \tag{1.6}$$

The initiation matrix π gives the probability of each state at the first step. The objective of HMM analysis is to reconstruct the hidden state sequence using the most likely model parameters $\lambda=(A, B, \pi)$ that describe the system.

Simple examples including coin tossing experiment and the urn-and-ball process were introduced to describe the HMM [121, 124]. Suppose that we have N urns, and each urn contains colored balls with M different fractions of colors. The proportion of color m in the i-th urn is $p_i(m)$ (i=1,2,…,N. m=1,2,…,M). Assume an experiment that a person can

randomly choose an urn, pick up one ball, and record its color. After the ball is replaced in the urn, then the new urn is selected and the experiment is repeated until time *T*. In this example, the observation sequence corresponds to the sequence of recorded color of balls, and the hidden state is the sequence of selected urns. With the trained model parameters, the urn selection sequence can be inferred from the observed sequence of colored balls.

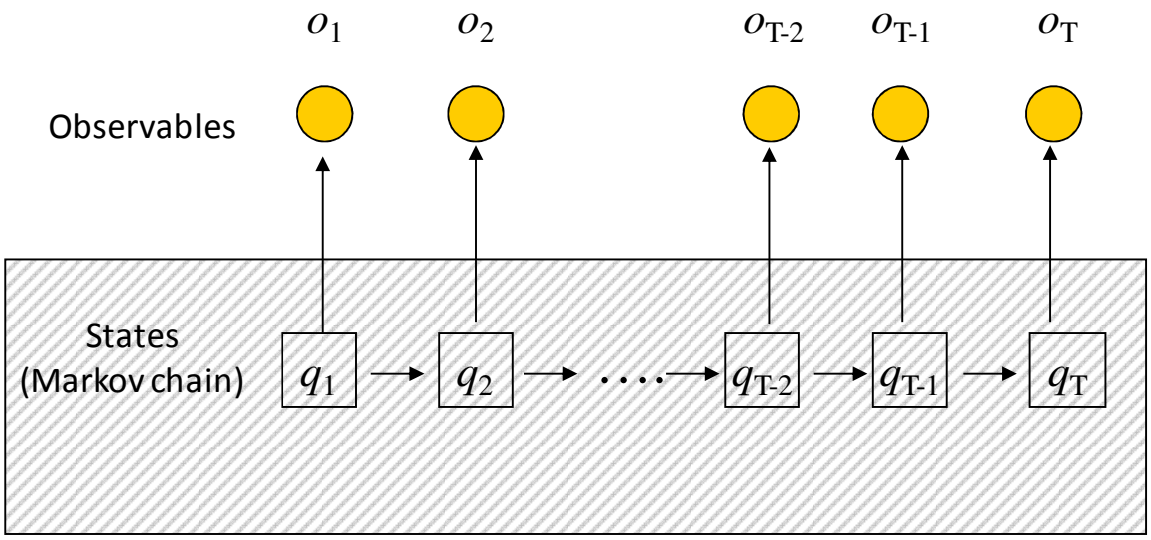

Figure 1.4 A typical HMM time series. A hidden state sequence q_t (t=1, 2, …, T-1, T) follows Markov property, and observables o_t (t=1, 2, …, T-1, T) distribute by emission probabilities at given state.

1.5.2 Algorithms of hidden Markov models

Three major objectives of the hidden Markov model are (1) to calculate the likelihood of observation given model parameters, (2) to adjust model parameters given observation to maximize the likelihood, (3) to reconstruct hidden state sequences, and (4) to select the most probable model. Several algorithms, including forward-backward algorithm [125], Baum-Welch (BW) algorithm [126, 127], and Viterbi algorithm [128, 129] have been developed to address major questions in HMM.

1.5.2.1 Calculating likelihood

The forward-backward algorithm enables calculation of the likelihood of observed data requiring fewer calculations. Given observed data O, state sequence Q, and model parameter λ, the likelihood $P(O|\lambda)$ is calculated by:

$$P(O \mid \lambda) = \sum_{all\ Q} P(O \mid Q, \lambda) P(Q \mid \lambda)$$
$$= \sum_{q_1, q_2, \cdots q_T} \pi_{q_1}\, b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T) \tag{1.7}$$

where π_q is the initiation probability of state q, $b_q(O_j)$ is the emission probability that o_j is observed from state q, and a_{ij} is the transition probability from state i to state j. If T observations are given and N states exist, calculating the likelihood of all possible state sequences requires $2TN^T$ multiplications, which would be an astronomical number even with a small number of states or observations. With 3 states and 1000 data points, the calculation requires $2 \cdot 1000 \cdot 3^{1000} \approx 10^{500}$ multiplications. However, the forward-backward algorithm does not require such huge number of calculations to determine the likelihood. The forward variable of state j at time t is defined by:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(o_t) \tag{1.8}$$

Therefore, forward variable $\alpha_t(j)$ is the joint probability that the current state is j with observed sequence $o_1, o_2, \ldots, o_t$ until time t given model parameter λ.

$$\alpha_t(j) = P(o_1, o_2, \cdots, o_t, q_t = j|\lambda) \tag{1.9}$$

If we calculate eq. (1.9) over all observed data, multiply by the transition probability and sum over N states, the resulting forward variable results in the likelihood of the observed data given the model parameters.

$$\sum_{j=1}^{N} \alpha_T(j) = \sum_{j=1}^{N} P(o_1, o_2, \cdots o_T, q_T = j|\lambda) = P(O|\lambda) \tag{1.10}$$

Compared to eq. (1.7) requiring $2TN^T$ multiplications, eq. (1.10) requires only NT multiplications and NT summations, which makes for a significantly faster calculation.

1.5.2.2 Parameter estimation

Model parameters including initiation, transition, and emission matrices need to be adjusted to maximize the posterior probability $P(\lambda|O)$. The posterior probability is the probability of certain model parameters λ, provided the observation O is obtained. The posterior probability can be expressed using the likelihood $P(O|\lambda)$ and the prior probability of model parameters $P(\lambda)$ as follows:

$$P(\lambda \mid O) = \frac{P(O \mid \lambda)P(\lambda)}{P(O)} = \frac{P(O \mid \lambda)P(\lambda)}{\sum_i P(O \mid \lambda_i)P(\lambda_i)} \tag{1.11}$$

Thus, maximization of the posterior probability is equivalent to the maximization of likelihood if the model parameters are assumed to be uniformly distributed.

The BW algorithm has been developed to maximize the likelihood of HMM processes [126, 127]. In the BW algorithm, a new variable, the backward variable, is introduced to describe posterior probabilities. A backward variable $\beta_t(i)$ is defined by eq. (1.12) and it represents a joint probability of state i at time t and observation sequence from time t+1 to T, $o_{t+1},o_{t+2},\ldots,o_T$ given model parameters.

$$\beta_t(i) = \sum_{i=1}^{N} a_{ij}\, b_j(o_{t+1})\beta_{t+1}(j) \tag{1.12}$$

Combining forward and backward variables, the estimated transition probability from state i to state j, $\bar{a}_{ij}$, can be calculated by eq. (1.13) given observed data and initial transition and emission matrices.

$$\bar{a}_{ij} = \frac{\text{Expected number of transitions from state } i \text{ to state } j}{\text{Expected number of staying at state } i}$$

$$= \frac{\sum_t P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)}{\sum_t P(q_t = s_i \mid O, \lambda)} = \frac{\sum_t \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_t \alpha_t(i)\beta_t(i)} \tag{1.13}$$

In a similar way, the expected emission probability of observing x at state i is estimated by:

$$\bar{b}_i(x) = \frac{\text{expected number of observing } x \text{ in state } i}{\text{expected number of staying at state } i} = \frac{\sum_{O_t = x} \alpha_t(i)\beta_t(i)}{\sum_t \alpha_t(i)\beta_t(i)} \tag{1.14}$$

These estimated transition and emission matrices have been mathematically proven to result in larger likelihood [130]. Therefore, the BW algorithm is a kind of expectation-maximization (EM) algorithm.[121]. Estimated transition and emission probabilities are used as initial parameters in the next estimation iteration and the iteration is repeated until a convergence criterion is reached.

Although quite fast, the traditional BW algorithm simultaneously requires a great deal of data for adequate training and is prone to trapping in inherent local maxima rather than converging to global maxima of likelihood [131, 132]. Simulated annealing has been used to escape from local maxima by cooling temperature functions and reducing energy functions which corresponds to likelihood [133]. In iterated local search algorithms, initial parameters are optimized until a local minimum is obtained, and the optimized parameters are replaced by neighborhoods in parameter space [134]. Those methods, however, require long running times.

Unfortunately, when physical constraints that govern distributions of collected data exist, BW algorithms must be modified accordingly, therefore further degrading their overall accuracy in finding a global solution to the statistical problem at hand. The introduction of physical constraints is one of the main advances of the work presented in this thesis. In order to address this issue, I introduced photophysical conditions to parameter estimation which constrain transition and emission matrices to be photophysically relevant and lead the fitting procedure to escape from local maxima and proceed toward global maxima. Using this scheme, the most probable number of states and the best photophysical model can be directly elucidated from experimental data. This will be discussed in more detail in chapter 3.

1.5.2.3 Hidden state reconstruction

The most probable state path can be reconstructed after all model parameters are trained to have maximum likelihood. Calculating the probability of every possible state sequence requires N^T calculations. One useful criteria in finding the optimal state sequence is to determine the most probable state individually at every data point [135].

The probability $\gamma_t(i)$ of state i at time t given the observation sequence O and model parameters λ is as follows:

$$\gamma_t(i) = P(q_t = i \mid O, \lambda) = \frac{P(O, q_t = i \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{1.15}$$

However, this criterion has a problem that eq. (1.15) determines the most probable state individually at every data point, not as a whole sequence. The compilation of the most probable state at individual time point does not always correspond to the most likely state sequence. Thus, the optimality of as a whole sequence should be estimated to determine the most probable sequence efficiently.

Dynamic programming for shortest path problem is a promising solution for finding the optimal state sequence [136]. Figure 1.3 shows a trellis structure for the shortest path problem. At each stage, the cost of transition from state i to state j is calculated, and the minimum cost is used to calculate the cost at the next stage. The cost of forbidden state transition is considered to be infinite. After all the cost calculations are done by reaching the artificial terminal, backtracking of the path with the minimum cost at every stage determines the shortest path.

Utilizing the short path problem for HMM, Viterbi algorithm has been developed to find the optimal state sequence [128, 129]. The cost in short path problem corresponds to the product of transition and emission probability. The Viterbi algorithm is the solution for reconstructing a state sequence which maximizes a posteriori probability $P(Q|O,\lambda)$. $P(Q|O,\lambda)$ denotes the probability of a state sequence with the given observation sequence and model parameters.

The first part of the Viterbi algorithm is to determine the most probable state at the beginning and ending. If the probability of the most probable path $\psi_t(i)$ that ends in state i at time t, and its probability with observation o_t is $\delta_t(j)$, then the next probable path and its probability can be found by multiplying transition and emission matrices as eq. (1.16).

$$\delta_t(j) = b_j(o_t) \max_i\left[\delta_{t-1}(i)a_{ij}\right]$$

$$\psi_t(i) = \operatorname{argmax}_i\left[\delta_{t-1}(i)a_{ij}\right] \quad (1.16)$$

where $\operatorname{argmax}_i\left[\delta_{t-1}(i)a_{ij}\right]$ is the state i that maximizes $\delta_{t-1}(i)a_{ij}$.

The most probable state path can be found by applying eq. (1.16) recursively from the end to the first data point. The Viterbi algorithm can be used to train model parameters by renewing transition and emission matrices using reconstructed state sequence. While its iteration is faster and computationally inexpensive, the likelihood may not converge to local or global maxima [137, 138]

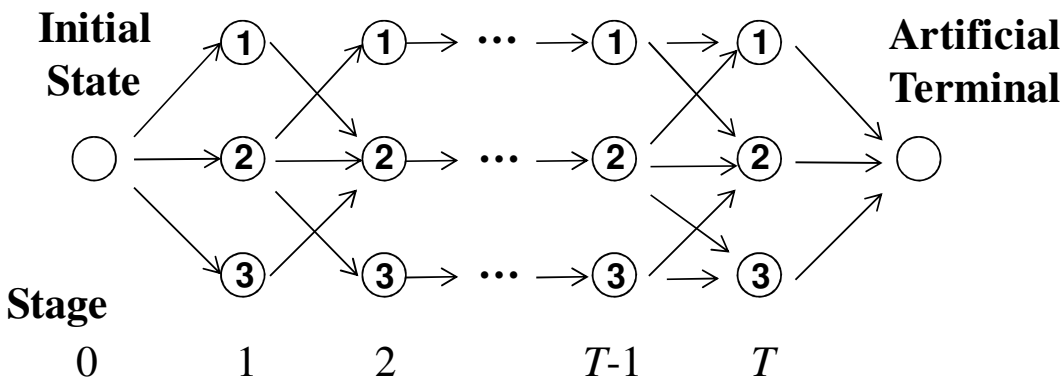

Figure 1.5 Transition graph for short path problem (adapted from [136]). The nodes correspond to states.

1.5.2.4 Model selection

Proper state sequence reconstruction requires determination of the true number of states, and thus makes evaluation of the most likely system dimension a key problem. However, adding more hidden states to model parameters will monotonically increase the likelihood of observed data and result in overfitting [139-142]. To avoid overfitting and best estimate the true number of states, several information criteria have been developed, including Akaike information (AIC)[143], Hannan-Quinn (HQC)[144], and Bayesian information (BIC) criteria [105, 145-147]. These criteria modify maximum likelihood estimates with penalizing terms based on the Laplace-Bayesian approximation of the likelihood and prior probabilities in the limit of large sample-size [148]. Among those criteria, BIC has been broadly used in determining image class of positron emission tomography and magnetic resonance images of patient brains [149], determining the number of Gaussians for curvature fitting in 2D shape classification [150], and explaining the effect of the number of species on the combination of ecosystem function [151]. Although direct, model-independent information theoretical approaches [105] may also work well especially when a kinetic model is inapplicable, even these model-independent methods rely on these same information criteria that may not consistently find the simplest and most likely model for short data sets. Furthermore, for kinetic models, HMM has enjoyed wide applicability and success [106, 152-154], suggesting that any information criteria improving accuracy, especially for short data sets may be of great utility. While BIC has been shown to provide a rigorous upper limit on the true number of states for an infinitely long sequence,[148] many important experimental data sets are far too short to satisfy this requirement. Therefore, in order to improve the performance

of modeling for Poissonian emitters, we introduced modifications to the conventional information criteria. In this thesis, chi-square probability is used as the information criteria for time-binned photon trajectory, and BIC is used in analyzing time-binned and photon-by-photon time trajectory.

1.5.3 Applications of HMM

Because the most probable state sequence can be extracted from complicated and noisy data using trained system parameters, HMM has been utilized in various fields. Originally referred as probabilistic functions of Markov chains, the basic theory of HMM was developed and introduced to describe hidden dynamics of ecological models and stock market behavior in late 1960s [125-127].

Automatic speech recognition, one of the major applications of HMM, has emerged in 1970s [121, 133, 153, 155, 156]. The acoustic signals of speech differ among individuals even though they speak the same word; even the same word repeatedly spoken by a single person has different signals. Therefore, interpretation of the unknown utterance with reference templates becomes inadequate, and consequently stochastic modeling is required for converting speech signals into meaningful words or sentences [135]. Since 1970s, HMM has been implemented to model decoding errors in which input speech waveform is translated to output representation [155]. HMM has been widely used in recognizing isolated words [157], word-by-word speech with random pause [158-160], and continuous speech with large vocabulary [161-164]. Speech signal processing based on HMM has enabled speech enhancement [165], speaker identification [165], and classification of environmental noise events such as car pass-bys and aircraft fly-overs [166].

One of the advantages of HMM is that the modeling ability is not limited by the distribution of observations [123]. Any discrete or continuous probability density function in the real world can be modeled including exponential [167], Gaussian [145, 168, 169], Weibull [170], Gamma and its mixture distributions [167, 169, 171-174]. Particularly, exponential family such as gamma and Erlang distributions are useful to model duration between events including human response [175], neural spike trains [167], and failure time of interconnects in integrated circuits [173]. Gaussian distribution is suited to describe additive noise in measurements including initial amount of the DNA molecules before PCR [176], nanopore channel current [168], and FRET efficiency [145].

The versatility in modeling various observation distributions has made HMM useful in biological signal analysis. HMM has been applied to build the most probable model describing the behavior of a single ion channel gating, by analyzing the single ion channel current [177-186]. Ion channels are large proteins which controls signal transmission and membrane potential across the lipid bilayer of the neuronal membrane [100]. Single channel ion currents are the function of underlying gating kinetics of ion channel, which can be measured by patch clamp techniques [100]. The ion channel current is very small (~femtoamperes) and may easily be buried in Gaussian or environmental noise. Stochastic analyses are required to idealize noisy current and extract gating state transitions and dwell times [187]. HMM-based methods have been successful in detecting idealized single ion channel current, short-lived events that are usually missed by low-pass filtering [185, 186], as well as reconstructing complicated signals in the presence of highly-expressed multiple ion channels [177, 181, 183]. HMMs have been applied to interpret neural spike trains to investigate the stimulus-response

relationship [188-193]. In these papers, the hidden states are neuronal responses to the stimuli input or different sample behavior modes, and the observations are spike firing rates recorded from cortical areas in monkeys or rats. The underlying process was assumed to be finite-state, time homogeneous Markov process. Spike trains processing using HMMs revealed the correlation between firing rate patterns and responses of cortical neurons to different tastes in awake rats [190], and enabled parameter estimations of UP-DOWN state dynamics in rats during slow-wave sleep [191].

Another application of HMMs in biology is biological sequence analysis of genes and proteins. Probabilistic modeling of gene sequence is useful in relating proteins and predicting those structures or properties [130]. One of the early applications of HMMs to sequence analysis was to model the population of G-C clusters in the mitochondrial DNA of yeast and humans, and to describe compositional heterogeneity of mammalian mitochondrial and bacteriophage DNA [194]. HMM-based modeling has been extensively used in aligning and examining similarities among protein sequences [195, 196], determining change points of DNA copy numbers [197-200], and gene prediction [201-204].

The major obstacle to single molecule analysis is the low signal-to-noise ratio induced by electronic, mechanical, and thermal noises. Due to the ability to extract model parameters in the presence of white or colored noise [205-207], HMMs have been used to model single molecule results with insufficient signal-to-noise ratio such as particle tracking [208, 209], single ionic current measurement [185], dwell time analysis [210], FRET analysis [145, 154, 211, 212], and simulation of single molecule fluorescence and kinetics [106, 213]. Although incompatible with state sequence reconstructions, non-

Markovian memory effects due to thermal fluctuation [214] can be quantified through correlation analysis [215].

The Markov-modulated Poisson process (MMPP), which is compatible with continuous HMM, is useful for modeling and analyzing burst signals [110, 111]. The MMPP is a double stochastic process in which arrival rates of Poisson process change according to the states of continuous Markov processes. The model parameters, including infinitesimal generator matrix for Markov processes and Poisson arrival rate matrix, can be converted back and forth into transition and emission matrices, which will be described later. The MMPP has been applied to model arrival processes in ATM-networks [112, 113], IP network management [114], and accidental long-range exposure to radioactivity [115].

Due to its ability to model telegraph and discontinuous signal processes, the MMPP has found wide application in analysis of photophysical processes. Photon arrival processes were simulated based on quantum jumps between ground singlet, excited singlet, and triplet states [216]. On- and off-period durations were estimated with maximized likelihood, assuming two-level system dynamics. Dynamics of complex formation and dissociation with Cu^{2+} ions were studied using fluorescence quenching of tetramethylrhodamine (TMR) [217]. Fluorescence emission decreases upon binding of Cu^{2+} ions to TMR-bipyridine conjugate, and the binding and dissociation constant of Cu^{2+} ions to bipyridine ligands were estimated from the fluorescence intensity time trace at different Cu^{2+} concentrations. The MMPP has also been employed to extract information from one-color photon trajectories [218]. Transition rates between two to

four fluorescent states with different intensities were estimated from simulated photons, and the number of states was determined by AIC and BIC.

1.6 Objectives of thesis

In this study, first I will show how to apply HMM to analyze the intensity trajectories from a single Poisson emitter and to modify the fitting algorithm to improve model-selection accuracy in Chapter 2. In Chapter 3, the development of photon-by-photon HMM (PbPHMM) will be discussed, including the theoretical aspects of PbPHMM. Photon time traces will be generated using HMM-based methods. Experimental and photophysical parameters will be adjusted to mimic realistic data, considering non-unity photon collection efficiency, nonzero noise level, or varying dark state quantum yield and lifetime. Fitting performance of PbPHMM will be evaluated by analyzing photon time traces at various conditions. In chapter 4, PbPHMM will be applied to analyze the experimental data collected from optically modulated silver nanodots and AcGFP.

1.7 References

1. Sevick-Muraca, E.M., G. Lopez, J.S. Reynolds, T.L. Troy, and C.L. Hutchinson, *F luorescence and Absorption Contrast Mechanisms for Biomedical Optical Imagin g Using Frequency-Domain Techniques.* Photochemistry and Photobiology, 1997. **66**(1): p. 55-64.

2. Mahmood, U. and R. Weissleder, *Near-Infrared Optical Imaging of Proteases in Cancer.* Molecular Cancer Therapeutics, 2003. **2**(5): p. 489-496.

3. Sokolov, K., M. Follen, J. Aaron, I. Pavlova, A. Malpica, R. Lotan, and R. Richar ds-Kortum, *Real-Time Vital Optical Imaging of Precancer Using Anti-Epidermal*

Growth Factor Receptor Antibodies Conjugated to Gold Nanoparticles. Cancer R esearch, 2003. **63**(9): p. 1999-2004.

4. Shah, K. and R. Weissleder, *Molecular Optical Imaging: Applications Leading to the Development of Present Day Therapeutics.* NeuroRX, 2005. **2**(2): p. 215-225.

5. Jain, P.K., X. Huang, I.H. El-Sayed, and M.A. El-Sayed, *Noble Metals on the Nan oscale: Optical and Photothermal Properties and Some Applications in Imaging, Sensing, Biology, and Medicine.* Accounts of Chemical Research, 2008. **41**(12): p. 1578-1586.

6. Kumar, A.T.N., E. Chung, S.B. Raymond, J.A.J.M. van de Water, K. Shah, D. Fuk umura, R.K. Jain, B.J. Bacskai, and D.A. Boas, *Feasibility of in vivo imaging of fl uorescent proteins using lifetime contrast.* Optics Letters, 2009. **34**(13): p. 2066-2 068.

7. Farrar, C.T., W.S. Kamoun, C.D. Ley, Y.R. Kim, S.J. Kwon, G. Dai, B.R. Rosen, E. di Tomaso, R.K. Jain, and A.G. Sorensen, *In vivo validation of MRI vessel cali ber index measurement methods with intravital optical microscopy in a U87 mous e brain tumor model.* Neuro-Oncology, 2010. **12**(4): p. 341-350.

8. Lawrence, J.B., R.H. Singer, and J.A. McNeil, *Interphase and Metaphase Resolut ion of Different Distances Within the Human Dystrophin Gene.* Science, 1990. **24 9**(4971): p. 928-932.

9. Weissleder, R., C.-H. Tung, U. Mahmood, and A. Bogdanov, *In vivo imaging of tu mors with protease-activated near-infrared fluorescent probes.* Nat Biotech, 1999. **17**(4): p. 375-378.

10. Baumgart, T., S.T. Hess, and W.W. Webb, *Imaging coexisting fluid domains in bio membrane models coupling curvature and line tension.* Nature, 2003. **425**(6960): p. 821-824.

11. Larson, D.R., W.R. Zipfel, R.M. Williams, S.W. Clark, M.P. Bruchez, F.W. Wise, and W.W. Webb, *Water-Soluble Quantum Dots for Multiphoton Fluorescence Ima ging in Vivo.* Science, 2003. **300**(5624): p. 1434-1436.

12. Gao, X., Y. Cui, R.M. Levenson, L.W.K. Chung, and S. Nie, *In vivo cancer targeti ng and imaging with semiconductor quantum dots.* Nat Biotech, 2004. **22**(8): p. 9 69-976.

13. Stroh, M., J.P. Zimmer, D.G. Duda, T.S. Levchenko, K.S. Cohen, E.B. Brown, D. T. Scadden, V.P. Torchilin, M.G. Bawendi, D. Fukumura, and R.K. Jain, *Quantum dots spectrally distinguish multiple species within the tumor milieu in vivo.* Nat M ed, 2005. **11**(6): p. 678-682.

14. Fukumura, D. and R.K. Jain, *Tumor microvasculature and microenvironment: Tar gets for anti-angiogenesis and normalization.* Microvascular Research, 2007. **74**(2–3): p. 72-84.

15. Popović, Z., W. Liu, V.P. Chauhan, J. Lee, C. Wong, A.B. Greytak, N. Insin, D.G. Nocera, D. Fukumura, R.K. Jain, and M.G. Bawendi, *A Nanoparticle Size Series f or In Vivo Fluorescence Imaging.* Angewandte Chemie, 2010. **122**(46): p. 8831-8 834.

16. Hell, S.W. and J. Wichmann, *Breaking the diffraction resolution limit by stimulate d emission: stimulated-emission-depletion fluorescence microscopy.* Optics letters, 1994. **19**(11): p. 780-782.

17. Hell, S.W. and M. Kroug, *Ground-state-depletion fluorscence microscopy: A conc ept for breaking the diffraction resolution limit.* Applied Physics B, 1995. **60**(5): p . 495-497.

18. Klar, T.A., S. Jakobs, M. Dyba, A. Egner, and S.W. Hell, *Fluorescence microscop y with diffraction resolution barrier broken by stimulated emission.* Proceedings o f the National Academy of Sciences, 2000. **97**(15): p. 8206-8210.

19. Betzig, E., G.H. Patterson, R. Sougrat, O.W. Lindwasser, S. Olenych, J.S. Bonifac ino, M.W. Davidson, J. Lippincott-Schwartz, and H.F. Hess, *Imaging Intracellular Fluorescent Proteins at Nanometer Resolution.* Science, 2006. **313**(5793): p. 164 2-1645.

20. Hess, S.T., T.P.K. Girirajan, and M.D. Mason, *Ultra-High Resolution Imaging by Fluorescence Photoactivation Localization Microscopy.* Biophysical Journal, 200 6. **91**(11): p. 4258-4272.

21. Rust, M.J., M. Bates, and X. Zhuang, *Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM).* Nat Meth, 2006. **3**(10): p. 793-796.

22. Eggeling, C., J. Widengren, R. Rigler, and C.A.M. Seidel, *Photostability of Fluore scent Dyes for Single-Molecule Spectroscopy: Mechanisms and Experimental Met*

hods for Estimating Photobleaching in Aqueous Solution, in *Applied Fluorescence in Chemistry, Biology and Medicine*. 1999, Springer Berlin Heidelberg. p. 193-240.

23. Levitus, M. and S. Ranjit, *Cyanine dyes in biophysical research: the photophysics of polymethine fluorescent dyes in biomolecular environments.* Quarterly Reviews of Biophysics, 2011. **44**(01): p. 123-151.

24. Michalet, X., F.F. Pinaud, L.A. Bentolila, J.M. Tsay, S. Doose, J.J. Li, G. Sundaresan, A.M. Wu, S.S. Gambhir, and S. Weiss, *Quantum Dots for Live Cells, in Vivo Imaging, and Diagnostics.* Science, 2005. **307**(5709): p. 538-544.

25. Alivisatos, A.P., *Semiconductor Clusters, Nanocrystals, and Quantum Dots.* Science, 1996. **271**(5251): p. 933-937.

26. Petty, J.T., J. Zheng, N.V. Hud, and R.M. Dickson, *DNA-Templated Ag Nanocluster Formation.* Journal of the American Chemical Society, 2004. **126**(16): p. 5207-5212.

27. Richards, C.I., S. Choi, J.-C. Hsiang, Y. Antoku, T. Vosch, A. Bongiorno, Y.-L. Tzeng, and R.M. Dickson, *Oligonucleotide-Stabilized Ag Nanocluster Fluorophores.* Journal of the American Chemical Society, 2008. **130**(15): p. 5038-5039.

28. Tsien, R.Y., *The green fluorescent protein.* Annu Rev Biochem, 1998. **67**(1): p. 509-544.

29. Wang, Y., J.Y.-J. Shyy, and S. Chien, *Fluorescence Proteins, Live-Cell Imaging, and Mechanobiology: Seeing Is Believing.* Annual Review of Biomedical Engineering, 2008. **10**(1): p. 1-38.

30. Bernard, J., L. Fleury, H. Talon, and M. Orrit, *Photon bunching in the fluorescence from single molecules: A probe for intersystem crossing.* The Journal of Chemical Physics, 1993. **98**(2): p. 850-859.

31. Yip, W.-T., D. Hu, J. Yu, D.A. Vanden Bout, and P.F. Barbara, *Classifying the Photophysical Dynamics of Single- and Multiple-Chromophoric Molecules by Single Molecule Spectroscopy.* The Journal of Physical Chemistry A, 1998. **102**(39): p. 7564-7575.

32. Efros, A.L. and M. Rosen, *Random Telegraph Signal in the Photoluminescence In tensity of a Single Quantum Dot.* Physical Review Letters, 1997. **78**(6): p. 1110-11 13.

33. Frantsuzov, P.A. and R.A. Marcus, *Explanation of quantum dot blinking without t he long-lived trap hypothesis.* Physical Review B, 2005. **72**(15): p. 155321.

34. Richards, C.I., J.-C. Hsiang, and R.M. Dickson, *Synchronously Amplified Fluores cence Image Recovery (SAFIRe).* The Journal of Physical Chemistry B, 2009. **114** (1): p. 660-665.

35. Richards, C.I., J.-C. Hsiang, D. Senapati, S. Patel, J. Yu, T. Vosch, and R.M. Dicks on, *Optically Modulated Fluorophores for Selective Fluorescence Signal Recover y.* Journal of the American Chemical Society, 2009. **131**(13): p. 4619-4621.

36. Valeur, B. and M.N. Berberan-Santos, *Molecular fluorescence: principles and app lications*. 2nd ed. 2012: John Wiley & Sons.

37. Zheng, Q., M.F. Juette, S. Jockusch, M.R. Wasserman, Z. Zhou, R.B. Altman, and S.C. Blanchard, *Ultra-stable organic fluorophores for single-molecule research.* Chemical Society Reviews, 2014. **43**(4): p. 1044-1056.

38. Sauer, M., J. Hofkens, and J. Enderlein, *Handbook of fluorescence spectroscopy a nd imaging : from single molecules to ensembles*. 2011, Weinheim: Wiley-VCH. i x, 281 p.

39. Yu, S.-J., M.-W. Kang, H.-C. Chang, K.-M. Chen, and Y.-C. Yu, *Bright Fluoresce nt Nanodiamonds: No Photobleaching and Low Cytotoxicity.* Journal of the Am erican Chemical Society, 2005. **127**(50): p. 17604-17605.

40. Horspool, W.M. and F. Lenci, *CRC handbook of organic photochemistry and phot obiology*. 2nd ed. 2004, Boca Raton: CRC Press.

41. Hsiang, J.-C., A.E. Jablonski, and R.M. Dickson, *Optically Modulated Fluorescen ce Bioimaging: Visualizing Obscured Fluorophores in High Background.* Acc. Ch em. Res., 2014. **47**(5): p. 1545-1554.

42. Richards, C.I., S. Choi, J.-C. Hsiang, Y. Antoku, T. Vosch, A. Bongiorno, Y.-L. Tz eng, and R.M. Dickson, *Oligonucleotide-stabilized Ag nanocluster fluorophores.* J

. Am. Chem. Soc., 2008. **130**(15): p. 5038-5039.

43. Vosch, T., Y. Antoku, J.-C. Hsiang, C.I. Richards, J.I. Gonzalez, and R.M. Dickso n, *Strongly emissive individual DNA-encapsulated Ag nanoclusters as single-mole cule fluorophores.* Proc. Natl. Acad. Sci. U. S. A., 2007. **104**(31): p. 12616-12621.

44. Nienhaus, K. and G. Ulrich Nienhaus, *Fluorescent proteins for live-cell imaging with super-resolution.* Chemical Society Reviews, 2014. **43**(4): p. 1088-1106.

45. Stennett, E.M.S., M.A. Ciuba, and M. Levitus, *Photophysical processes in single molecule organic fluorescent probes.* Chemical Society Reviews, 2014. **43**(4): p. 1 057-1075.

46. Dickson, R.M., A.B. Cubitt, R.Y. Tsien, and W.E. Moerner, *On/off blinking and s witching behavior of single molecules of green fluorescent protein.* Nature (Londo n), 1997. **388**(6640): p. 355-358.

47. Renn, A., J. Seelig, and V. Sandoghdar, *Oxygen-dependent photochemistry of fluo rescent dyes studied at the single molecule level.* Molecular Physics, 2006. **104**(3) : p. 409-414.

48. Aitken, C.E., R.A. Marshall, and J.D. Puglisi, *An oxygen scavenging system for i mprovement of dye stability in single-molecule fluorescence experiments.* Biophys ical Journal, 2008. **94**(5): p. 1826-1835.

49. Dave, R., D.S. Terry, J.B. Munro, and S.C. Blanchard, *Mitigating unwanted photo physical processes for improved single-molecule fluorescence imaging.* Biophysic al Journal, 2009. **96**(6): p. 2371-2381.

50. Heilemann, M., E. Margeat, R. Kasper, M. Sauer, and P. Tinnefeld, *Carbocyanine Dyes as Efficient Reversible Single-Molecule Optical Switch.* Journal of the Ameri can Chemical Society, 2005. **127**(11): p. 3801-3806.

51. Altman, R.B., Q. Zheng, Z. Zhou, D.S. Terry, J.D. Warren, and S.C. Blanchard, *E nhanced photostability of cyanine fluorophores across the visible spectrum.* Nat Meth, 2012. **9**(5): p. 428-429.

52. Zheng, Q., S. Jockusch, Z. Zhou, R.B. Altman, J.D. Warren, N.J. Turro, and S.C. Blanchard, *On the Mechanisms of Cyanine Fluorophore Photostabilization.* The J

ournal of Physical Chemistry Letters, 2012. **3**(16): p. 2200-2203.

53. Cook, R.J. and H.J. Kimble, *Possibility of Direct Observation of Quantum Jumps.* Physical Review Letters, 1985. **54**(10): p. 1023-1026.

54. Javanainen, J., *Possibility of quantum jumps in a three-level system.* Physical Revi ew A, 1986. **33**(3): p. 2121-2123.

55. Basche, T., S. Kummer, and C. Brauchle, *Direct spectroscopic observation of qua ntum jumps of a single molecule.* Nature, 1995. **373**(6510): p. 132-134.

56. Widengren, J., U. Mets, and R. Rigler, *Fluorescence correlation spectroscopy of t riplet states in solution: a theoretical and experimental study.* The Journal of Phys ical Chemistry, 1995. **99**(36): p. 13368-13379.

57. Ha, T., T. Enderle, D.S. Chemla, P.R. Selvin, and S. Weiss, *Quantum jumps of sin gle molecules at room temperature.* Chemical Physics Letters, 1997. **271**(1–3): p. 1-5.

58. Weston, K.D., P.J. Carson, H. Metiu, and S.K. Buratto, *Room-temperature fluores cence characteristics of single dye molecules adsorbed on a glass surface.* The Jo urnal of Chemical Physics, 1998. **109**(17): p. 7474-7485.

59. Xie, X.S. and J.K. Trautman, *OPTICAL STUDIES OF SINGLE MOLECULES AT ROOM TEMPERATURE.* Annual Review of Physical Chemistry, 1998. **49**(1): p. 441.

60. Garcia-Parajo, M.F., G.M.J. Segers-Nolten, J.A. Veerman, J. Greve, and N.F. Van Hulst, *Real-time light-driven dynamics of the fluorescence emission in single gree n fluorescent protein molecules.* Proceedings of the National Academy of Sciences of the United States of America, 2000. **97**(13): p. 7237-7242.

61. Yeow, E.K.L., S.M. Melnikov, T.D.M. Bell, F.C. De Schryver, and J. Hofkens, *Ch aracterizing the fluorescence intermittency and photobleaching kinetics of dye mo lecules immobilized on a glass surface.* Journal of Physical Chemistry A, 2006. **11 0**(5): p. 1726-1734.

62. Widengren, J., R. Rigler, and U. Mets, *Triplet-state monitoring by fluorescence co rrelation spectroscopy.* Journal of Fluorescence, 1994. **4**(3): p. 255-258.

63. Schwille, P., F.J. Meyer-Almes, and R. Rigler, *Dual-color fluorescence cross-corr elation spectroscopy for multicomponent diffusional analysis in solution.* 1997. **72** (4): p. 1878-1886.

64. Yip, W.-T., D. Hu, J. Yu, D.A. Vanden Bout, and P.F. Barbara, *Classifying the Pho tophysical Dynamics of Single- and Multiple-Chromophoric Molecules by Single Molecule Spectroscopy.* J. Phys. Chem. A, 1998. **102**(39): p. 7564-7575.

65. Nirmal, M., B.O. Dabbousi, M.G. Bawendi, J.J. Macklin, J.K. Trautman, T.D. Har ris, and L.E. Brus, *Fluorescence intermittency in single cadmium selenide nanocr ystals.* Nature, 1996. **383**(6603): p. 802-804.

66. Kuno, M., D.P. Fromm, S.T. Johnson, A. Gallagher, and D.J. Nesbitt, *Modeling di stributed kinetics in isolated semiconductor quantum dots.* Physical Review B, 20 03. **67**(12): p. 125304.

67. Frantsuzov, P.A., Volk, aacute, K. n, oacute, S., ndor, Jank, and Bolizs, *Model of F luorescence Intermittency of Single Colloidal Semiconductor Quantum Dots Usin g Multiple Recombination Centers.* Physical Review Letters, 2009. **103**(20): p. 20 7402.

68. Frantsuzov, P.A., S. Volkán-Kacsó, and B. Jankó, *Universality of the Fluorescence Intermittency in Nanoscale Systems: Experiment and Theory.* Nano Letters, 2012. **13**(2): p. 402-408.

69. Kuno, M., D.P. Fromm, H.F. Hamann, A. Gallagher, and D.J. Nesbitt, *"On"/"off" f luorescence intermittency of single semiconductor quantum dots.* Journal of Chem ical Physics, 2001. **115**(2): p. 1028-1040.

70. Moerner, W.E. and L. Kador, *Optical detection and spectroscopy of single molecu les in a solid.* Physical Review Letters, 1989. **62**(21): p. 2535-2538.

71. Orrit, M. and J. Bernard, *Single pentacene molecules detected by fluorescence exc itation in a p-terphenyl crystal.* Physical Review Letters, 1990. **65**(21): p. 2716-2719.

72. Uemura, S., M. Dorywalska, T.-H. Lee, H.D. Kim, J.D. Puglisi, and S. Chu, *Pepti de bond formation destabilizes Shine-Dalgarno interaction on the ribosome.* Natu re (London, U. K.), 2007. **446**(Copyright (C) 2012 American Chemical Society (A CS). All Rights Reserved.): p. 454-457.

73. Larson, M.H., W.J. Greenleaf, R. Landick, and S.M. Block, *Applied Force Reveal s Mechanistic and Energetic Details of Transcription Termination.* Cell, 2008. **13 2**(6): p. 971-982.

74. Kim, J., C.-Z. Zhang, X. Zhang, and T.A. Springer, *A mechanically stabilized rece ptor-ligand flex-bond important in the vasculature.* Nature, 2010. **466**(7309): p. 9 92-995.

75. Chodera, J.D., P. Elms, F. Noe, B. Keller, C.M. Kaiser, A. Ewall-Wice, S. Marqus ee, C. Bustamante, and N.S. Hinrichs, *Bayesian hidden Markov model analysis of single-molecule force spectroscopy: characterizing kinetics under measurement u ncertainty.* arXiv.org, e-Print Arch., Condensed Matter, 2011(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 1-12, arXiv:1108.14 30v1101 [cond-mat.stat-mech].

76. McKinney, S.A., A.-C. Déclais, D.M.J. Lilley, and T. Ha, *Structural dynamics of i ndividual Holliday junctions.* Nature Structural Biology, 2003. **10**(2): p. 93-97.

77. Myong, S., I. Rasnik, C. Joo, T.M. Lohman, and T. Ha, *Repetitive shuttling of a m otor protein on DNA.* Nature, 2005. **437**(7063): p. 1321-1325.

78. Harms, G.S., G. Orr, M. Montal, B.D. Thrall, S.D. Colson, and H.P. Lu, *Probing Conformational Changes of Gramicidin Ion Channels by Single-Molecule Patch-Clamp Fluorescence Microscopy.* Biophysical Journal, 2003. **85**(3): p. 1826-1838.

79. Sakmann, B. and E. Neher, *Single-channel recording*. 2009: Springer New York.

80. Colquhoun, D. and A.G. Hawkes, *On the Stochastic Properties of Single Ion Chan nels.* Proceedings of the Royal Society of London. Series B, Biological Sciences, 1981. **211**(1183): p. 205-235.

81. Rigler, R. and E. Elson, *Fluorescence correlation spectroscopy : theory and appli cations*. Springer series in chemical physics,. 2001, Berlin ; New York: Springer. x x, 487 p.

82. Magde, D., E. Elson, and W.W. Webb, *Thermodynamic Fluctuations in a Reacting System—Measurement by Fluorescence Correlation Spectroscopy.* Physical Revi ew Letters, 1972. **29**(11): p. 705-708.

83. Lu, H.P., L. Xun, and X.S. Xie, *Single-Molecule Enzymatic Dynamics.* Science, 1 998. **282**(5395): p. 1877-1882.

84. Kettling, U., A. Koltermann, P. Schwille, and M. Eigen, *Real-time enzyme kinetics monitored by dual-color fluorescence cross-correlation spectroscopy.* Proceeding s of the National Academy of Sciences of the United States of America, 1998. **95**(4): p. 1416-1420.

85. Haase, M., C.G. Hübner, E. Reuther, A. Herrmann, K. Müllen, and T. Basché, *Exp onential and Power-Law Kinetics in Single-Molecule Fluorescence Intermittency†* . The Journal of Physical Chemistry B, 2004. **108**(29): p. 10445-10450.

86. Widengren, J., J. Dapprich, and R. Rigler, *Fast interactions between Rh6G and d GTP in water studied by fluorescence correlation spectroscopy.* Chemical Physics , 1997. **216**(3): p. 417-426.

87. Haupts, U., S. Maiti, P. Schwille, and W.W. Webb, *Dynamics of fluorescence fluct uations in green fluorescent protein observed by fluorescence correlation spectros copy.* Proceedings of the National Academy of Sciences of the United States of A merica, 1998. **95**(23): p. 13573-13578.

88. Widengren, J., Ü. Mets, and R. Rigler, *Photodynamic properties of green fluoresc ent proteins investigated by fluorescence correlation spectroscopy.* Chemical Phys ics, 1999. **250**(2): p. 171-186.

89. Jablonski, A.E., R.B. Vegh, J.-C. Hsiang, B. Bommarius, Y.-C. Chen, K.M. Solnts ev, A.S. Bommarius, L.M. Tolbert, and R.M. Dickson, *Optically modulatable blue fluorescent proteins.* J. Am. Chem. Soc., 2013. **135**(44): p. 16410-16417.

90. Patel, S.A., M. Cozzuol, J.M. Hales, C.I. Richards, M. Sartin, J.-C. Hsiang, T. Vos ch, J.W. Perry, and R.M. Dickson, *Electron Transfer-Induced Blinking in Ag Nano dot Fluorescence.* J. Phys. Chem. C, 2009. **113**(47): p. 20264-20270.

91. Eggeling, C., C. Ringemann, R. Medda, G. Schwarzmann, K. Sandhoff, S. Polyak ova, V.N. Belov, B. Hein, C. von Middendorff, A. Schonle, and S.W. Hell, *Direct observation of the nanoscale dynamics of membrane lipids in a living cell.* Nature, 2009. **457**(7233): p. 1159-1162.

92. Eigen, M. and R. Rigler, *Sorting single molecules: application to diagnostics and evolutionary biotechnology.* Proceedings of the National Academy of Sciences, 19

94. **91**(13): p. 5740-5747.

93. Schenter, G.K., H.P. Lu, and X.S. Xie, *Statistical Analyses and Theoretical Model s of Single-Molecule Enzymatic Dynamics.* The Journal of Physical Chemistry A, 1999. **103**(49): p. 10477-10488.

94. Oleg, K. and B. Grégoire, *Fluorescence correlation spectroscopy: the technique a nd its applications.* Reports on Progress in Physics, 2002. **65**(2): p. 251.

95. Vosch, T., Y. Antoku, J.-C. Hsiang, C.I. Richards, J.I. Gonzalez, and R.M. Dickso n, *Strongly emissive individual DNA-encapsulated Ag nanoclusters as single-mole cule fluorophores.* Proceedings of the National Academy of Sciences, 2007. **104**(3 1): p. 12616-12621.

96. Gopich, I.V. and A. Szabo, *Single-Macromolecule Fluorescence Resonance Energ y Transfer and Free-Energy Profiles.* The Journal of Physical Chemistry B, 2003. **107**(21): p. 5058-5063.

97. Gopich, I. and A. Szabo, *Theory of photon statistics in single-molecule F[o-umlau t]rster resonance energy transfer.* The Journal of Chemical Physics, 2005. **122**(1): p. 014707.

98. Gopich, I.V. and A. Szabo, *Single-Molecule FRET with Diffusion and Conformati onal Dynamics.* The Journal of Physical Chemistry B, 2007. **111**(44): p. 12925-12 932.

99. Gopich, I.V. and A. Szabo, *Decoding the Pattern of Photon Colors in Single-Mole cule FRET.* The Journal of Physical Chemistry B, 2009. **113**(31): p. 10965-10973.

100. Sakmann, B. and E. Neher, *Single-channel recording*. 2nd ed. 1995, New York: Pl enum Press. xxii, 700 p.

101. Talaga, D.S., W.L. Lau, H. Roder, J. Tang, Y. Jia, W.F. DeGrado, and R.M. Hochst rasser, *Dynamics and folding of single two-stranded coiled-coil peptides studied b y fluorescent energy transfer confocal microscopy.* Proceedings of the National Ac ademy of Sciences, 2000. **97**(24): p. 13021-13026.

102. Chung, S.H. and R.A. Kennedy, *Forward-backward non-linear filtering technique for extracting small biological signals from noise.* Journal of Neuroscience Meth

ods, 1991. **40**(1): p. 71-86.

103. Haran, G., *Noise reduction in single-molecule fluorescence trajectories of folding proteins.* Chemical Physics, 2004. **307**(2–3): p. 137-145.

104. Rhoades, E., E. Gussakovsky, and G. Haran, *Watching proteins fold one molecule at a time.* Proceedings of the National Academy of Sciences, 2003. **100**(6): p. 319 7-3202.

105. Watkins, L.P. and H. Yang, *Detection of Intensity Change Points in Time-Resolved Single-Molecule Measurements.* The Journal of Physical Chemistry B, 2005. **109**(1): p. 617-628.

106. Messina, T.C., H. Kim, J.T. Giurleo, and D.S. Talaga, *Hidden Markov Model Anal ysis of Multichromophore Photobleaching.* The Journal of Physical Chemistry B, 2006. **110**(33): p. 16366-16376.

107. Bockenhauer, S., A. Fürstenberg, X.J. Yao, B.K. Kobilka, and W.E. Moerner, *Conf ormational Dynamics of Single G Protein-Coupled Receptors in Solution.* The Jou rnal of Physical Chemistry B, 2011. **115**(45): p. 13328-13338.

108. Schmidt, R., C. Krasselt, C. Göhler, and C. von Borczyskowski, *The Fluorescence Intermittency for Quantum Dots Is Not Power-Law Distributed: A Luminescence Intensity Resolved Approach.* ACS Nano, 2014. **8**(4): p. 3506-3521.

109. Colquhoun, D. and F.J. Sigworth, *Fitting and Statistical Analysis of Single-Chann el Records*, in *Single-Channel Recording*, B. Sakmann and E. Neher, Editors. 199 5, Springer US. p. 483-587.

110. K.S, M.-H., *A fitting algorithm for Markov-modulated poisson processes having t wo arrival rates.* European Journal of Operational Research, 1987. **29**(3): p. 370-3 77.

111. Rydén, T., *Parameter estimation for Markov modulated poisson processes.* Comm unications in Statistics. Stochastic Models, 1994. **10**(4): p. 795-829.

112. Kawashima, K. and H. Saito, *Teletraffic issues in ATM networks.* Computer Netw orks and ISDN Systems, 1990. **20**(1–5): p. 369-375.

113. Kang, S.H., K. Yong Han, D.K. Sung, and B.D. Choi, *An application of Markovia n arrival process (MAP) to modeling superposed ATM cell streams.* Communicati ons, IEEE Transactions on, 2002. **50**(4): p. 633-642.

114. Salvador, P., A. Pacheco, and R. Valadas, *Modeling IP traffic: joint characterizati on of packet arrivals and packet sizes using BMAPs.* Computer Networks, 2004. **4 4**(3): p. 335-352.

115. Ramesh, N.I., *Statistical analysis on markov-modulated poisson processes.* Enviro nmetrics, 1995. **6**(2): p. 165-179.

116. Kampen, N.G.v., *Stochastic processes in physics and chemistry*. 3rd ed. North-Ho lland personal library. 2007, Amsterdam ; Boston ; London: Elsevier. xvi, 463 p.

117. Mahnke, R., J. Kaupuzs, and I.O. Lubashevskyi, *Physics of stochastic processes : how randomness acts in time*. Physics textbook. 2009, Weinheim, Chichester: Wil ey-VCH ;John Wiley distributor. xvii, 430 p.

118. Vanzi, F., C. Broggio, L. Sacconi, and F.S. Pavone, *Lac repressor hinge flexibility and DNA looping: single molecule kinetics by tethered particle motion.* Nucleic A cids Research, 2006. **34**(12): p. 3409-3420.

119. Gration, K.A.F., J.J. Lambert, R.L. Ramsey, R.P. Rand, and P.N.R. Usherwood, *Cl osure of membrane channels gated by glutamate receptors may be a two-step proc ess.* Nature, 1982. **295**(5850): p. 599-601.

120. Hamill, O.P., A. Marty, E. Neher, B. Sakmann, and F.J. Sigworth, *Improved patch-clamp techniques for high-resolution current recording from cells and cell-free me mbrane patches.* Pflügers Archiv, 1981. **391**(2): p. 85-100.

121. Rabiner, L.R., *A tutorial on hidden Markov models and selected applications in sp eech recognition.* Proceedings of the IEEE, 1989. **77**(2): p. 257-286.

122. Ephraim, Y. and N. Merhav, *Hidden Markov processes.* Information Theory, IEEE Transactions on, 2002. **48**(6): p. 1518-1569.

123. Bilmes, J.A., *What HMMs can do.* IEICE TRANSACTIONS on Information and Systems, 2006. **89**(3): p. 869-891.

124. Poritz, A. *Hidden Markov models: a guided tour*. in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*. 1988.

125. Baum, L.E. and J. Eagon, *An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology.* Bull. Amer. Math. Soc, 1967. **73**(3): p. 360-363.

126. Baum, L.E., T. Petrie, G. Soules, and N. Weiss, *A Maximization Technique Occurr ing in the Statistical Analysis of Probabilistic Functions of Markov Chains.* The A nnals of Mathematical Statistics, 1970. **41**(1): p. 164-171.

127. Baum, L.E., *An inequality and associated maximization technique in statistical est imation for probabilistic functions of Markov processes.* Inequalities, 1972. **3**: p. 1 -8.

128. Viterbi, A.J., *Error bounds for convolutional codes and an asymptotically optimu m decoding algorithm.* Information Theory, IEEE Transactions on, 1967. **13**(2): p. 260-269.

129. Forney, G.D., Jr., *The viterbi algorithm.* Proceedings of the IEEE, 1973. **61**(3): p. 268-278.

130. Durbin, R., *Biological sequence analysis : probalistic models of proteins and nucl eic acids*. 1998, Cambridge, UK New York: Cambridge University Press. xi, 356 p.

131. Baldi, P. and Y. Chauvin, *Smooth on-line learning algorithms for hidden Markov models.* Neural Computation, 1994. **6**(2): p. 307.

132. Shatkay, H. and L.P. Kaelbling, *Learning topological maps with weak local odom etric information.* Proceedings of IJCAI'97, 1997: p. 920-929.

133. Paul, D.B. *Training of HMM recognizers by simulated annealing*. in *Acoustics, Sp eech, and Signal Processing, IEEE International Conference on ICASSP '85.* 1985 .

134. Martin, O., S.W. Otto, and E.W. Felten, *Large-step Markov chains for the TSP inc orporating local search heuristics.* Operations Research Letters, 1992. **11**(4): p. 2 19-224.

135. Rabiner, L.R. and B.H. Juang, *Fundamentals of speech recognition*. Prentice Hall signal processing series. 1993, Englewood Cliffs, N.J.: PTR Prentice Hall. xxxv, 5 07 p.

136. Bertsekas, D.P., *Dynamic programming and optimal control*. 1995, Belmont, Mas s.: Athena Scientific.

137. Rodríguez, L. and I. Torres, *Comparative Study of the Baum-Welch and Viterbi Tr aining Algorithms Applied to Read and Spontaneous Speech Recognition*, in *Patte rn Recognition and Image Analysis*, F. Perales, et al., Editors. 2003, Springer Berli n Heidelberg. p. 847-857.

138. Humburg, P., D. Bulger, and G. Stone, *Parameter estimation for robust HMM ana lysis of ChIP-chip data.* BMC Bioinformatics, 2008. **9**(Copyright (C) 2012 U.S. N ational Library of Medicine.): p. 343.

139. Lanterman, A.D. *Minimum description length understanding of infrared scenes*. 1 998.

140. Stoica, P. and Y. Selen, *Model-order selection: a review of information criterion r ules.* Signal Processing Magazine, IEEE, 2004. **21**(4): p. 36-47.

141. Calvet, L.E. and A. Fisher, *Multifractal volatility : theory, forecasting, and pricing* . Academic Press advanced finance series. 2008, Amsterdam ; Boston: Academic Press. xiii, 258 p.

142. Theodoridis, S. and R. Chellappa, *Academic Press Library in Signal Processing, V olume 3 : Array and Statistical Signal Processing*, ed. S. Theodoridis and R. Chell appa. 2013, Jordon Hill, GBR: Academic Press.

143. Akaike, H., *A new look at the statistical model identification.* Automatic Control, IEEE Transactions on, 1974. **19**(6): p. 716-723.

144. Hannan, E.J. and B.G. Quinn, *The Determination of the Order of an Autoregressio n.* Journal of the Royal Statistical Society. Series B (Methodological), 1979. **41**(2) : p. 190-195.

145. McKinney, S.A., C. Joo, and T. Ha, *Analysis of Single-Molecule FRET Trajectorie s Using Hidden Markov Modeling.* Biophysical Journal, 2006. **91**(5): p. 1941-195

1.

146. Schwarz, G., *Estimating the Dimension of a Model.* The Annals of Statistics, 1978 . **6**(2): p. 461-464.

147. Zhang, K., H. Chang, A. Fu, A.P. Alivisatos, and H. Yang, *Continuous Distributio n of Emission States from Single CdSe/ZnS Quantum Dots.* Nano Letters, 2006. **6**(4): p. 843-847.

148. Lanterman, A.D., *Schwarz, Wallace, and Rissanen: Intertwining Themes in Theori es of Model Selection.* International Statistical Review, 2001. **69**(2): p. 185-212.

149. Liang, Z., R.J. Jaszczak, and R.E. Coleman, *Parameter estimation of finite mixtur es using the EM algorithm and information criteria with application to medical i mage processing.* Nuclear Science, IEEE Transactions on, 1992. **39**(4): p. 1126-11 33.

150. Bicego, M. and V. Murino, *Investigating hidden Markov models' capabilities in 2 D shape classification.* IEEE Transactions on Pattern Analysis and Machine Intelli gence, 2004. **26**(2): p. 281-286.

151. Hector, A. and R. Bagchi, *Biodiversity and ecosystem multifunctionality.* Nature, 2 007. **448**(7150): p. 188-190.

152. Xiaolin, L., M. Parizeau, and R. Plamondon, *Training hidden Markov models with multiple observations-a combinatorial method.* IEEE Transactions on Pattern Ana lysis and Machine Intelligence, 2000. **22**(4): p. 371-377.

153. Levinson, S.E., L.R. Rabiner, and M.M. Sondhi, *An introduction to the applicatio n of the theory of probabilistic functions of a Markov process to automatic speech recognition.* The Bell System Technical Journal, 1983. **62**(4): p. 1035-1074.

154. Lee, T.-H., *Extracting Kinetics Information from Single-Molecule Fluorescence R esonance Energy Transfer Data Using Hidden Markov Models.* The Journal of Ph ysical Chemistry B, 2009. **113**(33): p. 11535-11542.

155. Bahl, L.R. and F. Jelinek, *Decoding for channels with insertions, deletions, and su bstitutions with applications to speech recognition.* Information Theory, IEEE Tra nsactions on, 1975. **21**(4): p. 404-411.

156. Bahl, L.R., F. Jelinek, and R. Mercer, *A Maximum Likelihood Approach to Contin uous Speech Recognition.* Pattern Analysis and Machine Intelligence, IEEE Trans actions on, 1983. **PAMI-5**(2): p. 179-190.

157. Rabiner, L.R., B.H. Juang, S.E. Levinson, and M.M. Sondhi, *Recognition of Isolat ed Digits Using Hidden Markov Models With Continuous Mixture Densities.* AT& T Technical Journal, 1985. **64**(6): p. 1211-1234.

158. Rabiner, L.R., J.G. Wilpon, and B.-H. Juang, *A segmental k-means training proce dure for connected word recognition.* AT&T Technical Journal, 1986. **65**(3): p. 21-31.

159. Rabiner, L.R., J.G. Wilpon, and B.H. Juang, *A model-based connected-digit recog nition system using either hidden Markov models or templates.* Computer Speech & Language, 1986. **1**(2): p. 167-197.

160. Biing-Hwang, J. and L. Rabiner, *The segmental <e1>K</e1>-means algorithm fo r estimating parameters of hidden Markov models.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 1990. **38**(9): p. 1639-1641.

161. Ney, H., R. Haeb-Umbach, B.H. Tran, and M. Oerder. *Improvements in beam sear ch for 10000-word continuous speech recognition*. in *Acoustics, Speech, and Sign al Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. 1992.

162. Giachin, E.P., C.H. Lee, L.R. Rabiner, A.E. Rosenberg, and R. Pieraccini, *On the use of inter-word context-dependent units for word juncture modeling.* Computer Speech & Language, 1992. **6**(3): p. 197-213.

163. Huang, X.D. and M.A. Jack, *Semi-continuous hidden Markov models for speech s ignals.* Computer Speech & Language, 1989. **3**(3): p. 239-251.

164. Bahl, L.R., P.V. de Souza, P.S. Gopalakrishnan, D. Nahamoo, and M.A. Picheny. *Robust methods for using context-dependent features and models in a continuous speech recognizer*. in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94 ., 1994 IEEE International Conference on*. 1994.

165. Ephraim, Y., *Statistical-model-based speech enhancement systems.* Proceedings of the IEEE, 1992. **80**(10): p. 1526-1555.

166. Couvreur, C., V. Fontaine, P. Gaunard, and C.G. Mubikangiey, *Automatic Classifi cation of Environmental Noise Events by Hidden Markov Models.* Applied Acoust ics, 1998. **54**(3): p. 187-206.

167. Tokdar, S., P.Y. Xi, R.C. Kelly, and R.E. Kass, *Detection of bursts in extracellular spike trains using hidden semi-Markov point process models.* J Comput Neurosci, 2010. **29**(1-2): p. 203-212.

168. Churbanov, A. and S. Winters-Hilt, *Implementing EM and Viterbi algorithms for Hidden Markov Model in linear memory.* BMC Bioinformatics, 2008. **9**(Copyrigh t (C) 2012 U.S. National Library of Medicine.): p. 224.

169. Xi, L., Y. Fondufe-Mittendorf, L. Xia, J. Flatow, J. Widom, and J.-P. Wang, *Predic ting nucleosome positioning using a duration Hidden Markov Model.* BMC bioinf ormatics, 2010. **11**(1): p. 346.

170. Titman, A., *Estimating parametric semi-Markov models from panel data using ph ase-type approximations.* Statistics and Computing, 2014. **24**(2): p. 155-164.

171. Felsenstein, J. and G.A. Churchill, *A Hidden Markov Model approach to variation among sites in rate of evolution.* Molecular Biology and Evolution, 1996. **13**(1): p. 93-104.

172. Lin, T.W. and G.N. Reeke, *A Continuous Entropy Rate Estimator for Spike Trains Using a K-Means-Based Context Tree.* Neural Computation, 2010. **22**(4): p. 998-1 024.

173. Dwyer, V.M., *Analysis of multistate models for electromigration failure.* Journal o f Applied Physics, 2010. **107**(Copyright (C) 2012 American Chemical Society (A CS). All Rights Reserved.): p. 033709/033701-033709/033712.

174. Deng, L. and D.F. Moore, *Composite likelihood modeling of neighboring site corr elations of DNA sequence substitution rates.* Stat Appl Genet Mol Biol, 2009. **8**(C opyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. A rticle 6.

175. Fan, M., D. Gravem, D.M. Cooper, and D.J. Patterson, *Augmenting gesture recog nition with erlang-cox models to identify neurological disorders in premature babi es*, in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. 2012, ACM: Pittsburgh, Pennsylvania. p. 411-420.

176. Lalam, N., *A quantitative approach for polymerase chain reactions based on a hid den Markov model.* J Math Biol, 2009. **59**(4): p. 517-533.

177. Horn, R. and K. Lange, *Estimating kinetic constants from single channel data.* Bi ophysical Journal, 1983. **43**(2): p. 207-223.

178. Chung, S.H., J.B. Moore, L. Xia, L.S. Premkumar, and P.W. Gage, *Characterizati on of Single Channel Currents Using Digital Signal Processing Techniques Based on Hidden Markov Models.* Philosophical Transactions: Biological Sciences, 199 0. **329**(1254): p. 265-285.

179. Crouzy, S.C. and F.J. Sigworth, *Yet another approach to the dwell-time omission p roblem of single-channel analysis.* Biophysical Journal, 1990. **58**(3): p. 731-743.

180. Fredkin, D.R. and J.A. Rice, *Maximum Likelihood Estimation and Identification Directly from Single-Channel Recordings.* Proceedings of the Royal Society of Lo ndon. Series B: Biological Sciences, 1992. **249**(1325): p. 125-132.

181. Qin, F., A. Auerbach, and F. Sachs, *Estimating single-channel kinetic parameters f rom idealized patch-clamp data containing missed events.* Biophysical Journal, 19 96. **70**(1): p. 264-280.

182. Qin, F., A. Auerbach, and F. Sachs, *A Direct Optimization Approach to Hidden M arkov Modeling for Single Channel Kinetics.* Biophysical Journal, 2000. **79**(4): p. 1915-1927.

183. Qin, F., A. Auerbach, and F. Sachs, *Hidden Markov Modeling for Single Channel Kinetics with Filtering and Correlated Noise.* Biophysical Journal, 2000. **79**(4): p. 1928-1944.

184. Gunst, M.C.M.d., H.R. Kunsch, and J.G. Schouten, *Statistical Analysis of Ion Cha nnel Data Using Hidden Markov Models with Correlated State-Dependent Noise and Filtering.* Journal of the American Statistical Association, 2001. **96**(455): p. 8 05-815.

185. Qin, F., *Restoration of Single-Channel Currents Using the Segmental k-Means Me thod Based on Hidden Markov Modeling.* Biophysical Journal, 2004. **86**(3): p. 14 88-1501.

186. Nayak, T.K. and S.K. Sikdar, *Time-Dependent Molecular Memory in Single Volta ge-Gated Sodium Channel.* Journal of Membrane Biology, 2007. **219**(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 19-36.

187. Fredkin, D.R. and J.A. Rice, *Bayesian restoration of single-channel patch clamp r ecordings.* Biometrics, 1992. **48**(2): p. 427-448.

188. Abeles, M., H. Bergman, I. Gat, I. Meilijson, E. Seidemann, N. Tishby, and E. Vaa dia, *Cortical Activity Flips among Quasi-Stationary States.* Proceedings of the Na tional Academy of Sciences of the United States of America, 1995. **92**(19): p. 861 6-8620.

189. Seidemann, E., I. Meilijson, M. Abeles, H. Bergman, and E. Vaadia, *Simultaneous ly recorded single units in the frontal cortex go through sequences of discrete and stable states in monkeys performing a delayed localization task.* Journal of Neuros cience, 1996. **16**(2): p. 752-768.

190. Jones, L.M., A. Fontanini, B.F. Sadacca, P. Miller, and D.B. Katz, *Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles.* Proceedings of t he National Academy of Sciences, 2007. **104**(47): p. 18772-18777.

191. Chen, Z., S. Vijayan, R. Barbieri, M.A. Wilson, and E.N. Brown, *Discrete-and co ntinuous-time probabilistic models and algorithms for inferring neuronal UP and DOWN states.* Neural computation, 2009. **21**(7): p. 1797-1862.

192. Tokdar, S., P. Xi, R. Kelly, and R. Kass, *Detection of bursts in extracellular spike trains using hidden semi-Markov point process models.* Journal of Computational Neuroscience, 2010. **29**(1-2): p. 203-212.

193. Escola, S., A. Fontanini, D. Katz, and L. Paninski, *Hidden Markov models for the stimulus-response relationships of multistate neural systems.* Neural Comput, 201 1. **23**(5): p. 1071-1132.

194. Churchill, G.A., *Stochastic models for heterogeneous DNA sequences.* Bull Math Biol, 1989. **51**(1): p. 79-94.

195. Krogh, A., M. Brown, I.S. Mian, K. Sjolander, and D. Haussler, *Hidden Markov m odels in computational biology. Applications to protein modeling.* J Mol Biol, 199 4. **235**(5): p. 1501-1531.

196. Becker, E., A. Cotillard, V. Meyer, H. Madaoui, and R. Guerois, *HMM-Kalign: a t ool for generating sub-optimal HMM alignments.* Bioinformatics, 2007. **23**(Copyr ight (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 3095-3097.

197. Rueda, O.M. and R. Diaz-Uriarte, *Flexible and accurate detection of genomic cop y-number changes from aCGH.* PLoS Comput. Biol., 2007. **3**(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 1115-1122.

198. Stjernqvist, S., T. Ryden, M. Skoeld, and J. Staaf, *Continuous-index hidden Mark ov modeling of array CGH copy number data.* Bioinformatics, 2007. **23**(Copyrigh t (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 1006-101 4.

199. Rueda, O.M. and R. Diaz-Uriarte, *RJaCGH: Bayesian analysis of aCGH arrays f or detecting copy number changes and recurrent regions.* Bioinformatics, 2009. **2 5**(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 1959-1960.

200. Stjernqvist, S. and T. Ryden, *A continuous-index hidden Markov jump process for modeling DNA copy number data.* Biostatistics, 2009. **10**(Copyright (C) 2012 U.S . National Library of Medicine.): p. 773-778.

201. Lukashin, A.V. and M. Borodovsky, *GeneMark.hmm: New solutions for gene findi ng.* Nucleic Acids Research, 1998. **26**(4): p. 1107-1115.

202. Bergman, N.H., K.D. Passalacqua, P.C. Hanna, and Z.S. Qin, *Operon prediction f or sequenced bacterial genomes without experimental information.* Applied and E nvironment Microbiology, 2007. **73**(Copyright (C) 2012 American Chemical Soci ety (ACS). All Rights Reserved.): p. 846-854.

203. Brent, M.R., *How does eukaryotic gene prediction work?* Nature Biotechnology, 2 007. **25**(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reser ved.): p. 883-885.

204. Brown, D.P., N. Krishnamurthy, and K. Sjolander, *Automated protein subfamily id entification and classification.* PLoS Comput. Biol., 2007. **3**(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 1526-1538.

205. Sigworth, F.J., D.W. Urry, and K.U. Prasad, *Open channel noise. III. High-resolut*

ion recordings show rapid current fluctuations in gramicidin A and four chemical analogues. Biophysical Journal, 1987. **52**(6): p. 1055-1064.

206. Venkataramanan, L., R. Kuc, and F.J. Sigworth, *Identification of hidden Markov models for ion channel currents. II. State-dependent excess noise.* Signal Processi ng, IEEE Transactions on, 1998. **46**(7): p. 1916-1929.

207. Venkataramanan, L., J.L. Walsh, R. Kuc, and F.J. Sigworth, *Identification of hidde n Markov models for ion channel currents. I. Colored background noise.* Signal Pr ocessing, IEEE Transactions on, 1998. **46**(7): p. 1901-1915.

208. Beausang, J.F., C. Zurla, C. Manzo, D. Dunlap, L. Finzi, and P.C. Nelson, *DNA L ooping Kinetics Analyzed Using Diffusive Hidden Markov Model.* Biophysical Jou rnal, 2007. **92**(8): p. L64-L66.

209. Smith, D.A., W. Steffen, R.M. Simmons, and J. Sleep, *Hidden-Markov Methods f or the Analysis of Single-Molecule Actomyosin Displacement Data: The Variance-Hidden-Markov Method.* Biophysical Journal, 2001. **81**(5): p. 2795-2816.

210. Milescu, L.S., A. Yildiz, P.R. Selvin, and F. Sachs, *Extracting Dwell Time Sequen ces from Processive Molecular Motor Data.* Biophysical Journal, 2006. **91**(9): p. 3135-3150.

211. Liu, Y., J. Park, K.A. Dahmen, Y.R. Chemla, and T. Ha, *A Comparative Study of Multivariate and Univariate Hidden Markov Modelings in Time-Binned Single-M olecule FRET Data Analysis.* The Journal of Physical Chemistry B, 2010. **114**(16) : p. 5386-5403.

212. Uphoff, S., K. Gryte, G. Evans, and A.N. Kapanidis, *Improved Temporal Resoluti on and Linked Hidden Markov Modeling for Switchable Single-Molecule FRET.* C hemPhysChem, 2011. **12**(3): p. 571-579.

213. Andrec, M., R.M. Levy, and D.S. Talaga, *Direct Determination of Kinetic Rates fr om Single-Molecule Photon Arrival Trajectories Using Hidden Markov Models.* T he Journal of Physical Chemistry A, 2003. **107**(38): p. 7454-7464.

214. Talaga, D.S., *Markov processes in single molecule fluorescence.* Current Opinion in Colloid & Interface Science, 2007. **12**(6): p. 285-296.

215. Hu, D., R. Liu, X. Zeng, S. Kaplan, and H.P. Lu, *Fluctuating Two-State Light Har vesting in a Photosynthetic Membrane.* The Journal of Physical Chemistry C, 200 7. **111**(25): p. 8948-8956.

216. Burzykowski, T., J. Szubiakowski, and T. Rydén, *Analysis of photon count data fr om single-molecule fluorescence experiments.* Chemical Physics, 2003. **288**(2-3): p. 291-307.

217. Jäger, M., A. Kiel, D.-P. Herten, and F.A. Hamprecht, *Analysis of Single-Molecule Fluorescence Spectroscopic Data with a Markov-Modulated Poisson Process.* Ch emPhysChem, 2009. **10**(14): p. 2486-2495.

218. Hajdziona, M. and A. Molski, *Maximum likelihood-based analysis of single-mole cule photon arrival trajectories.* The Journal of Chemical Physics, 2011. **134**(5): p . 054112.

CHAPTER 2

FITTING TIME-BINNED DATA

2.1 Introduction

Time resolved analysis of fluorescence intensity trajectories details the photophysical behavior of single molecules [1, 2] and conformational changes of biomolecules [3, 4]. Since the fluorescence intensity is recorded as the number of photons detected within given time window, information on underlying dynamics may be lost if the binning time is longer than the time scale of the dynamics. Usually, fluorescence intensity is assumed to be Poissonian in most developed analyses. The Poisson assumption is valid for intensity traces from photophysical processes, which are faster than photon detection. However if the rate of photophysical processes including dark state relaxation is comparable to or slower than photon detection, the assumption is not sufficient to extract accurate parameters [5]. Fitting quality is even more aggravated by real experimental conditions. One must also account for environmental noise in all analyses that purport to reveal true photophysical dynamics

Due to its inherent flexibility, HMM can solve those problems to decode the true underlying dynamics. HMMs describe a system of interest with three model parameters - the transition, emission, and initiation matrices. The transition matrix is a matrix of which element a_{ij} is a conditional probability of a Markov chain from state i to state j. The emission matrix is a matrix of conditional probabilities that the current value is observed, given the current state. The initiation matrix is the probability of each state at the first data point. HMMs have turned out to be useful in modeling the temporal structure of a

sequence of unobserved states which account for different distributions of observed variables. The versatility of a HMM allows the modeling of any probability distribution in the real world, with enough data points [6].

The key algorithm for training model parameters is the Baum-Welch algorithm [7]. However, training iterations have shown that Baum-Welch suffers from iterations too often being trapped in local likelihood maxima. At this point, the likelihood of observed data is locally maximized and the iteration cannot escape from it to find the global maximum. Local maxima problems cause the likelihood not to be fully maximized, resulting in the wrong number of hidden states being identified. In this chapter, I will describe how to modify the training algorithm in HMMs for Poisson-distributed intensities. Data sets of varying length will be generated by pseudo-randomly generated transition and emission matrices exhibiting experimentally relevant, Poisson-distributed noise. We will fit the data sets by Baum-Welch and modified algorithms, using several types of criteria to determine the "true" number of states and compare to known values of simulated data sets. Effects of trajectory length and robustness relative to choice of initial conditions indicate that chi-square probability and localization error (LE) are crucial to proper state reconstructions.

2.2 Methods

Simulating typical single molecule fluorescence trajectories, 10,000 data sets of varying length with pseudo-randomly varying system parameters were generated in Matlab software (R2008a, Mathworks). Although emission levels were randomly chosen to match appropriately binned experimental data, our lone deviation from truly random

choices was that staying in the same state for the subsequent step was defined to be more probable than transition to any other individual state. The probability of staying in a given state could, however, be much less than 0.5. Obviously, the sum of each transition matrix row was constrained to unity, but the sum of each column varied widely due to the pseudo-random transition probabilities, yielding a wide range of state populations and transition probabilities in the simulated data traces. The emission levels (i.e. simulated intensity range) varied from 0 to 150 counts per bin. Emission matrices had levels of random mean value (fixed for each trajectory), subject to Poissonian sampling noise as determined by the total number of data points and state weight in each trajectory. The number of hidden states was varied from two to six, with 2000 unique data sets for each number of states. For simulated data having a given number of states, the number of data points per trajectory was increased by 200 for every 80 data sets. Thus the largest data sets have 5000 data points. In this notation, two states correspond, for example, to one bright level and one background, or "off" level. Only one emission level per state was allowed.

Baum-Welch training was performed by the standard programs included in Matlab. New algorithms resulted from the modification of the Baum-Welch algorithm using chi-square-minimized Poissonian fits and chi-square probability-determined relative weighting to determine the number of hidden states. The determined "correct" number of hidden states was extracted from the quality of the fits as follows: For each given number of states, after Baum-Welch training, the trained emission matrix is fitted to a Poisson distribution, and its initial weight is determined by the Viterbi algorithm. The emission level histogram is reconstructed by summing the properly weighted Poissonian emission

levels, each of which is fit to the best Poisson emission level through chi-square minimization. Chi-square is calculated between real histogram (simulated or experimental data) and the reconstructed one. The Poisson-constrained emission matrix and transition matrix are used as the initial system parameters for the next step. The iteration is continued until the global chi-square is minimized for a given number of states.

Although each level is fit with chi-square minimization to a Poisson distribution of intensities, two different criteria are used to determine the overall goodness of fit between the actual data and the reconstructed data – BIC and chi-square probability. BIC is calculated by

$$BIC = Loglikelihood - \left(\frac{d}{2}\right)\ln N_p \tag{2.1}$$

in which d is the degree-of-freedom of parameters to be trained when S is the number of states. The transition, emission, and initiation matrices are trained during BW algorithm. The transition matrix is an $S \times S$ matrix, and each element is the transition probability to be trained. One restriction per each row exists that the sum of each row should be one, which makes the degree-of-freedom of the transition matrix $S(S\text{-}1)$. The emission matrices of S different states are Poisson distributions, which are defined by an average intensity per state. The initiation matrix is a column vector of which elements are the probability of S different states at the beginning with one restriction. Therefore, the total degree-of-freedom d is $S(S\text{-}1) + S + S\text{-}1 = S^2 + S\text{-}1$. N_p is the number of data points. Chi-square between observed value X_i and expected value μ_i is defined by eq. (2).

$$\chi_n^2 = \sum_{i=1}^{n}\left(\frac{X_i - \mu_i}{\sigma_i}\right)^2 \tag{2.2}$$

in which n is the number of observables and σ_i is the standard deviation associated with the uncertainty in X_i.[8] Conversely, a chi-square variable, t, is governed by the probability distribution function.

$$probability = \frac{t^{n/2-1}e^{-t/2}}{2^{n/2}\Gamma^{(n/2)}}, \quad t = \chi_n^2 \tag{2.3}$$

Here, X_i is the probability of observable i in the experimental emission matrix, and μ_i is that in the Poisson-constrained emission matrix. For a given number of states, the training is stopped when the value of chi-square between the experimental and reconstructed histograms is minimized. Once a minimum for a given number of states is obtained, the number of states is changed by one and the global chi-square is again minimized. Since the degrees of freedom are related to the number of states, we used the integrated area of the chi-square distribution at the calculated chi-square value instead of the value itself. In this process, X_i is the number of occurrences of observable i in the real histogram, and μ_i is that in the Poisson-reconstructed histogram. This provides a method for comparing goodness-of-fit for different numbers of hidden states.[9]

Especially for short data sets, the above process often overfits the number of states, so a penalizing term giving a statistical measure of level distinguishability was incorporated by checking the overlaps of all emission matrix curves by localization error (LE). LE is defined by [10, 11]

$$\left\langle (\Delta x)^2 \right\rangle = \frac{\sigma^2}{N} \tag{2.4}$$

In optical localization experiments, Δx is the LE, σ is the standard deviation of point-spread function, and N is the number of collected photons. N and σ are replaced by the integrated area of the emission matrix of an individual state (its weight, or the number

of observations in that state) and the range of data values, respectively. When Δx is smaller than the difference between the mean values of two curves, overlapping curves were deleted, and the fitting process is performed again using the remaining curves as the initial parameters. Typically the fitting regenerates the overfitted results, with overlapping distributions that are indistinguishable by localization error. Therefore we terminated the iterations when the overlap appears. The previous best-fit number of states is then used as the best global fit. We denote the new algorithms modified by Poisson fit as PB when BIC is used to determine the dimension and PC when chi-square probability is used. In the case that LE is applied, the algorithms are represented as PBL and PCL, respectively.

2.3 Results and discussion

Although quite fast, the traditional Baum-Welch algorithm simultaneously requires a great deal of data for adequate training and is prone to trapping in local maxima and overfitting [12, 13]. Figure 2.1A illustrates a typical result of getting trapped in local maxima. During the training iteration, the log-likelihood monotonically increases as shown in Figure 2.1D (open circles). However, due to the lack of physical constraints and robust stopping point of the iterations, the resulting emission matrix curve is noisy and has indistinctly defined levels.

For time correlated single photon counting data, emission intensities should be Poisson-distributed. Constraining Baum-Welch to be physically reasonable demonstrates the advantage of the Poisson modified Baum-Welch algorithm (Figure 2.1). A simulated data set with 3800 data points and 5 hidden states was generated based on an emission

matrix consisting of 5 Poisson distributions. Compared to the emission matrix estimated by the Baum-Welch algorithm, the shape and peak position of the emission matrix fit by the Poisson modified Baum-Welch algorithm was much closer to the original emission matrix (shown in black solid lines) that was used in generating the data set. The overall emission level histogram of the simulated data set and that of the Baum-Welch and Poisson-constrained Baum-Welch reconstructions are nearly identical, but the Poisson constraints significantly improve the individual intensity distributions while simultaneously providing a clear maximum in the likelihood as a stopping point (Figure 2.1D). Such Poisson constraints on the emission levels enable more robust and more physically meaningful fits with better-defined stopping points.

Even with intensity levels constrained to be Poisson-distributed, however, the maximum likelihood method often overfits the data. For example, Figure 2.2A shows three curves with slightly different mean values being fitted to a single emissive level near 120. The number of data points in each distribution, however, is quite small, suggesting that the three curves may not be significantly different, but instead are consistent with a single distribution. Rather than eliminating overlaps visually, LE was introduced as a statistic to tie the actual weight of each state to the precision with which the distribution center can be determined. Using the standard deviation for the appropriate Poisson distribution, Figure 2.2B, for example, shows the advantage of using LE. Using this statistic, the curves in Figure 2.2A around the emission value of 120 were determined to be consistent with the single curve centered at 120 in Figure 2.2B. An approximation for asymmetric distributions, localization error works very well for higher intensities (>~20 counts/bin), where the differences between Poisson and Gaussian

distributions are relatively small. This method still gives good results for low intensities and is readily adapted for other common distributions, if necessary. Together this gives a meaningful method to determine the best fit for a given number of states. Comparing goodness-of-fit for different numbers of states, however, demands inclusion of additional criteria.

Figure 2.3 shows the histogram of fitting results from 1250 data sets generated using Poisson emission matrices with randomly assigned mean values. Both BIC and Chi-square probability (Equation 3) were compared as criteria for goodness-of-fit when varying the number of hidden states. Both incorporate more penalizing terms as more states are included to yield better-defined stopping points than maximum likelihood alone. Furthermore, the incorporation of LE significantly reduced the tendency of PB and PC to overfit the number of states, especially for short data sets. The number of hidden states was consequently estimated by the PBL (Figures 2.3A, C) and the PCL algorithms (Figures 2.3B, D), respectively, for a large number of simulated data sets. Initial parameters were determined in two ways. First, we used built-in peak finding codes in Matlab to identify initial emission levels from the histogram of each data set. Alternatively, we set the initial levels by eye from the histogram. The first method is automatic and much faster than the second one. However, frequently, the automated method did a poor job in predicting emission levels, especially when levels had close average values or for low numbers of counts per bin. Therefore, the accuracies of fitting results by manual initiation (Figures 2.3C, D) are much higher than that by automatic initiation (Figure 2.3A, B). Interestingly, Figures 2.3A and 2.3B demonstrate that in the case of automatic initiation, the determined “true” number of states by the PBL algorithm

is larger than that from the PCL algorithm. These results are an example of the property of BIC, i.e., BIC predicts the maximum number of probable dimensions in the limit of infinitely long data sets.[14, 15]

Not surprisingly, the fitting performance of all algorithms is highly dependent upon initial parameters, but the PCL algorithm appears least sensitive to poor initial guesses (Figure 2.4). Additionally, the algorithm to automatically generate a new emission matrix with one more or one less state did not work as well as did manual input. When we manually input a suspected level in every simulation, all Poissonian modified algorithms including PB, PBL, PC, and PCL algorithms tended to predict the correct dimension more frequently.

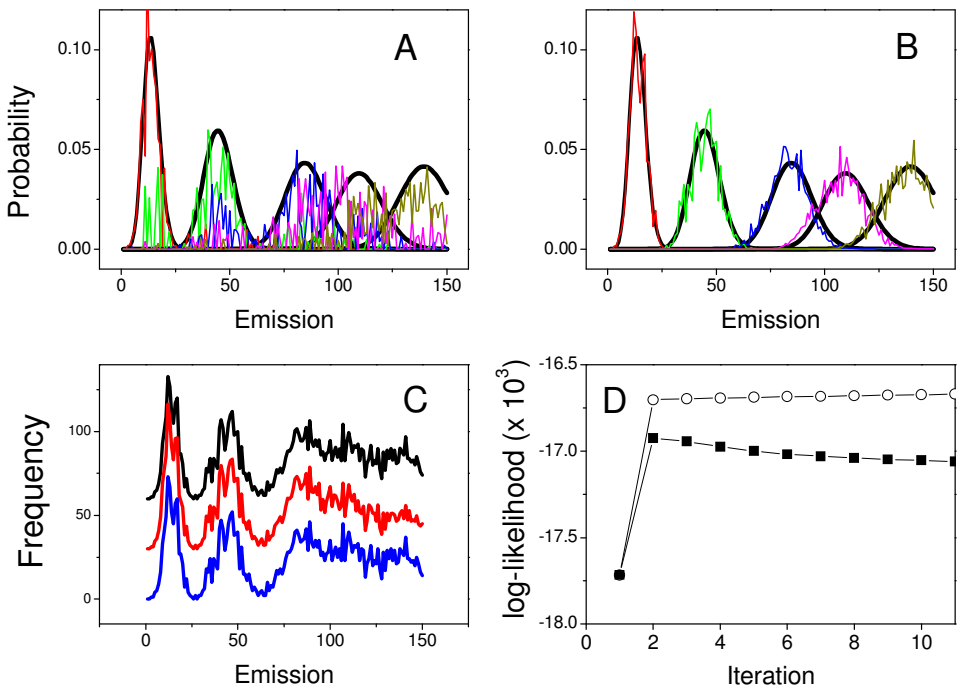

Figure 2.1 Comparison of traditional and Poisson-modified Baum-Welch algorithms. Emission matrices were trained by Baum-Welch (A) and Poisson-modified Baum-Welch (B) algorithms. Histogram (C) of a data set which has 3600 data points and 5 hidden states (▬, top) and reconstructed histograms from Baum-Welch (▬, middle) and Poisson-modified Baum-Welch (▬, bottom) algorithm. Histograms from reconstructed data are vertically offset from the simulated intensity histograms for clarity. (D) Log-likelihood from Baum-Welch (–○–) and Poissonian-modified Baum-Welch (–■–) algorithms of the same data set during the training iteration.

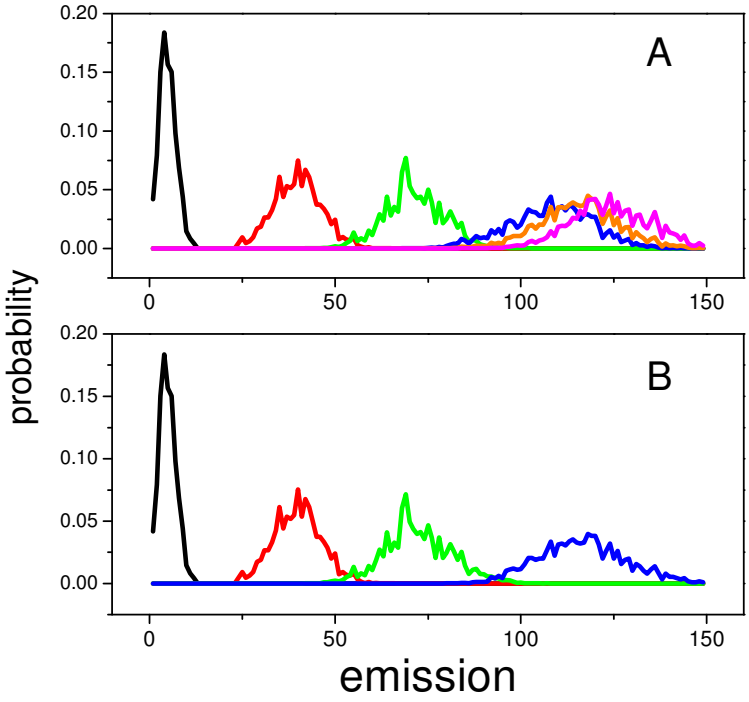

Figure 2.2 Modification by Localization Error (LE). Three curves at around 120 counts/bin in (A) are consolidated into one curve in (B).

A significant fraction of incorrect fitting results for all algorithms arises from the shorter data sets. The dependence of accuracy on the length of 6-state data sequence is shown in Figure 2.4C. A solid black line in this figure means the proportion of data sets which were generated using 6-state emission matrices and can be considered truly to have 6 hidden states (by localization error). The fraction is lower than 50% when the number of data points is smaller than 1000. This result can be explained in two ways. First, if we have too few data points, the system or molecule being measured may not access every possible state. Second, poorly sampled states have very large localization errors. Therefore, two low-occupancy adjacent curves in the emission matrix are likely to overlap, and two states are then consistent with a single level. In such cases, the fitting result tends to be smaller than the real answer. For these reasons, LE informs the setting of proper experimental conditions such as data collection time, bin width, or incident laser intensity. The accuracies of the PB and PC algorithms decrease with decreasing number of data points. However, the PCL algorithm appears largely unaffected by the N_p, relative to the PB, PBL, and PC algorithms. As shown in Figures 2.4B~F, the accuracies of the PCL algorithm is larger than 60% even if the number of data points is smaller than 1000. These results illustrate the robustness of the PCL algorithm, suggesting great utility for short data sets like those that plague single molecule studies.

Figure 2.5 shows all fitting results and the percentage of correct answers from five different conditions; Baum-Welch algorithm with BIC and the Poisson-modified algorithms using BIC and chi-square probability with and without LE. The error bars were determined by the standard deviation of 40 sets, each including 50 fitting results. The PCL algorithm (solid pattern) shows the best performance by t-test with 95% confidence interval.

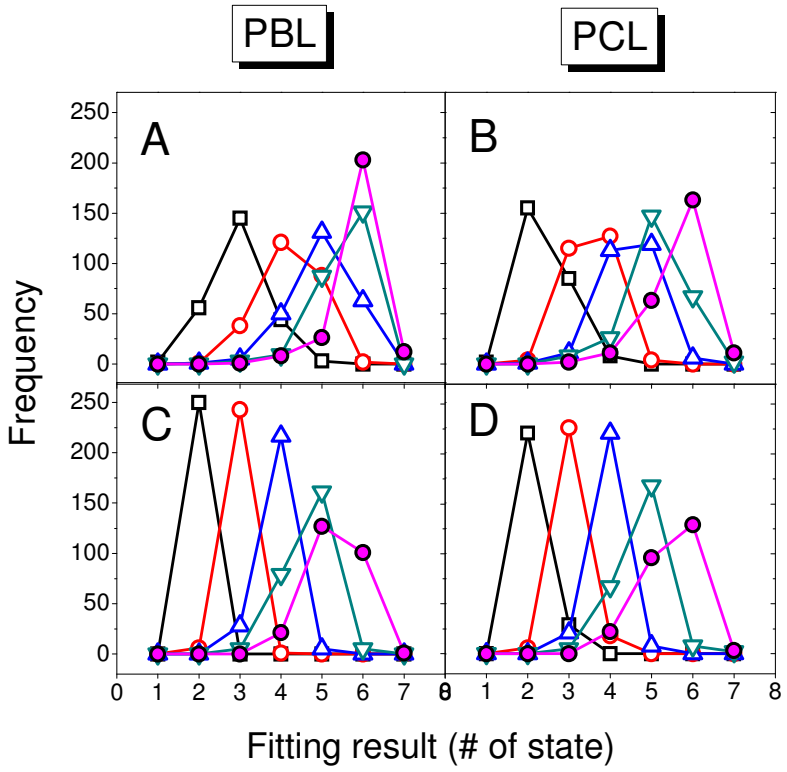

Figure 2.3 Histograms of fitting results calculated by PBL (A, C) and PCL (B, D) algorithms. Each data set was simulated to have 2 (), 3(), 4(), 5(), or 6() hidden states. The number of data points was varied from 200 to 5000. The initial parameters for top (A, B) and bottom panels (C, D) were acquired automatically and manually respectively.

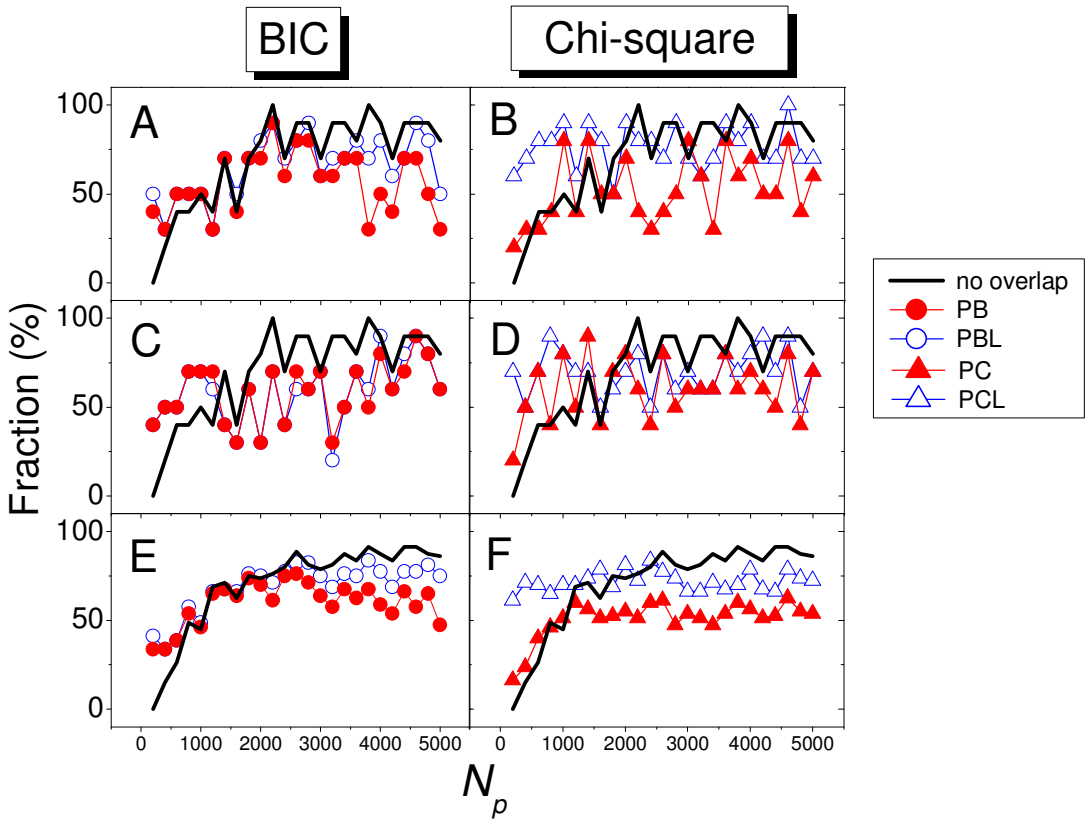

Figure 2.4 The effect of N_p on inherent number of states and accuracies of the four algorithms in analyzing 6-state data sets. 1250 data sets (A-D) and 10000 data sets (E, F) were generated based on 6-state emission matrices. Initial parameters were defined by an automatic peak finding algorithm (A, B, E, F) or manually (C, D). Solid lines show the proportion of 6-state data sets that have no overlap of curves in emission matrices. The accuracies were calculated by PB( ), PBL( ), PC( ), and PCL( ) algorithms.

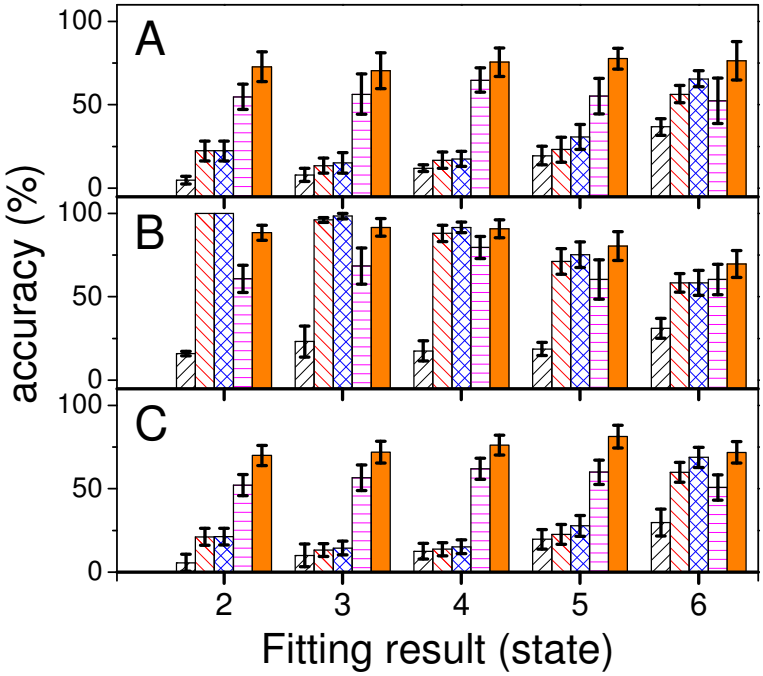

Figure 2.5 Accuracy bar plots of 5 kinds of criteria, BIC with Baum-Welch algorithm (▨), PB (▨), PC (▨), PBL (▨), and PCL (▨) algorithms. 1250 data sets were analyzed using automatically (A) and manually (B) defined initial parameters. (C) 10000 data sets were also computed with automatic initial parameters. Error bars show the standard deviation calculated from 40 sets of 50 accuracy results.

2.4 Conclusion

By modifying the Baum-Welch algorithm to introduce chi-square probability and localization error, we have improved HMM performance both in determining the dimensions of unknown systems and in robustness even if the intensity trajectory is very short, or if poor initial conditions are used. In these common experimental situations, the newly generated PCL algorithm can outperform even BIC with localization error for hidden Markov analysis of short trajectories.

2.5 References

1. Hofkens, J., M. Maus, T. Gensch, T. Vosch, M. Cotlet, F. Köhn, A. Herrmann, K. Müllen, and F. De Schryver, *Probing photophysical processes in individual multic hromophoric dendrimers by single-molecule spectroscopy.* Journal of the America n Chemical Society, 2000. **122**(38): p. 9278-9288.

2. Hernando, J., d.S.M. van, D.E.M.H.P. van, M. Sauer, M.F. Garcia-Parajo, and H.N .F. van, *Excitonic Behavior of Rhodamine Dimers: A Single-Molecule Study.* The J ournal of Physical Chemistry A, 2003. **107**(Copyright (C) 2012 American Chemic al Society (ACS). All Rights Reserved.): p. 43-52.

3. Bockenhauer, S., A. Fürstenberg, X.J. Yao, B.K. Kobilka, and W.E. Moerner, *Conf ormational Dynamics of Single G Protein-Coupled Receptors in Solution.* The Jou rnal of Physical Chemistry B, 2011. **115**(45): p. 13328-13338.

4. Kask, P., K. Palo, D. Ullmann, and K. Gall, *Fluorescence-Intensity Distribution A nalysis and Its Application in Biomolecular Detection Technology.* Proceedings of the National Academy of Sciences of the United States of America, 1999. **96**(24): p. 13756-13761.

5. Wang, J. and P. Wolynes, *Intermittency of Single Molecule Reaction Dynamics in Fluctuating Environments.* Physical Review Letters, 1995. **74**(21): p. 4317.

6. Bilmes, J., *What HMMs can do.* 2002.

7. Baum, L.E., *An inequality and associated maximization technique in statistical est imation for probabilistic functions of Markov processes.* Inequalities, 1972. **3**: p. 1 -8.

8. Bevington, P.R. and D.K. Robinson, *Data reduction and error analysis for the phy sical sciences*. 2nd ed. 1992, New York: McGraw-Hill. xvii, 328 p.

9. Barnes, J.W., *Statistical analysis for engineers and scientists : a computer-based approach*. 1994, New York: McGraw-Hill. xxvii, 396 p.

10. Ghosh, R.N. and W.W. Webb, *Automated detection and tracking of individual and clustered cell surface low density lipoprotein receptor molecules.* Biophysical Jou rnal, 1994. **66**(5): p. 1301-1318.

11. Thompson, R.E., D.R. Larson, and W.W. Webb, *Precise Nanometer Localization Analysis for Individual Fluorescent Probes.* Biophysical Journal, 2002. **82**(5): p. 2 775-2783.

12. Baldi, P. and Y. Chauvin, *Smooth on-line learning algorithms for hidden Markov models.* Neural Computation, 1994. **6**(2): p. 307.

13. Shatkay, H. and L.P. Kaelbling, *Learning topological maps with weak local odom etric information.* Proceedings of IJCAI'97, 1997: p. 920-929.

14. Watkins, L.P. and H. Yang, *Detection of Intensity Change Points in Time-Resolved Single-Molecule Measurements.* The Journal of Physical Chemistry B, 2005. **109**(1): p. 617-628.

15. Leroux, B.G., *Consistent Estimation of a Mixing Distribution.* The Annals of Stati stics, 1992. **20**(3): p. 1350-1360.

CHAPTER 3

PHOTON-BY-PHOTON HIDDEN MARKOV MODEL

3.1 Introduction

Time resolved analysis of fluorescence intensity trajectories can be used to understand the photophysical behavior of single molecules [1, 2] and conformational changes of biomolecules [3, 4]. Because an intensity is, for the purposes of this thesis, the number of photons detected within a given time window, information on underlying dynamics may be lost if the binning time is longer than the time scale of the dynamics. It is also important to consider the distribution of photons per unit time. This distribution is often assumed to be Poissonian. The Poisson assumption is valid for intensity traces arising from photophysical processes faster than photon detection. However, if the rate of photophysical processes including dark state relaxation is comparable or slower than photon detection, the assumption of Poisson statistics is not sufficient to extract accurate parameters [5]. Fitting quality is further deteriorated by real experimental conditions, such as environmental noise, which also should be accounted for.

The Hidden Markov Model (HMM) has been used to model the temporal structure of a sequence of unobserved states, such as speech representation, ion channel gating, protein conformation, and photophysical processes, which account for different distributions of observed variables [6-11]. For the case of photophysical transitions between discrete states, it is well suited to study the photophysical model of single fluorophores under excitation. HMMs describe the dynamics of a system of states and transitions by using three parameters: a transition, emission, and initiation matrix. The

transition matrix describes the conditional probability between states. The emission matrix is a matrix of conditional probabilities that a current value is observed, given a current state. The initiation matrix corresponds to the probabilities of each state at the first data point. The versatility of a HMM allows for the modeling of any probability distribution in the real world, provided sufficient sampling is obtained [12].

HMM-based analysis of time-binned trajectories has been used in modeling hidden dynamics in FRET data [10, 13-15], counting the number of fluorescent molecules [16], idealizing single ion channel current for modeling gating [17-21], and describing conformational dynamics of protein using force spectroscopy [22-26]. However, HMM analysis of time-binned data has several intrinsic weaknesses. HMM suffers from a loss of temporal information when faster than binning time. Additionally, extracting rate constants of state transition from transition matrix requires the solution of a master equation. The master equation completely describes the time evolution of a system modeled by transitions between exactly two of many discrete states, and becomes extremely and problematically complex in the case of three or more hidden states [9, 27, 28]. Many published studies of photophysical fluorescent state transitions are based on Poisson statistics. However, experimental detection efficiencies of < 100% lead to random photon loss and thus non-Poisson distribution of intensity counts.

In the HMM method, the rate of underlying process is extracted from trained model parameters including transition, emission, and initiation matrices $\lambda=(A, B, \pi)$, which are defined in Section 1.5.1. The key algorithm for training model parameters is the Baum-Welch (BW) algorithm, a numerical technique reliant upon multiple iterations [29]. It is possible that training by BW algorithm may lead parameters to violate physical

constraints of specific applications such as grammatical constraints in linguistic applications [30, 31], photon count distributions from a single Poisson emitter [32], and detailed balance in transition matrices [24, 33]. Additionally, through repeated iterations, training process may become trapped in a local maximum, unable to escape to the global maximum. If trapped, likelihood may not be fully maximized and result in the wrong number of hidden states. In this chapter, I will describe how I have solved these problems by developing a photon-by-photon HMM (PbPHMM), which is a photophysically-relevant algorithm to analyze simulated photon traces. Additionally, the effect of missed state transitions and background noise on model parameters will be examined, and correlations between estimated model parameters and photophysical parameters will be formulated.

3.2 Theory

3.2.1 Photon-by-photon HMM (PbPHMM)

Although HMM analysis can extract photophysical parameters from fluorescence intensity trajectories, time-binning averages out dynamics faster than the bin-time, and short bin-times cause slower calculations. Therefore, to keep all information and improve computational efficiency I developed PbPHMM. In this algorithm, photon waiting times are observation sequence values instead of intensities per constant time bin.

In the ideal case, every emitted photon is detected and a signal pulse is generated only by a photon detection event. A photon is emitted when an electronic relaxation from an excited singlet state (S_1) to ground photophysical state (S_0) takes place in Figure 3.1. Relaxation via intersystem crossing to a triplet dark state ($Dark_1$) does not result in photon emission. Therefore, there is no direct observed evidence about the dark state.

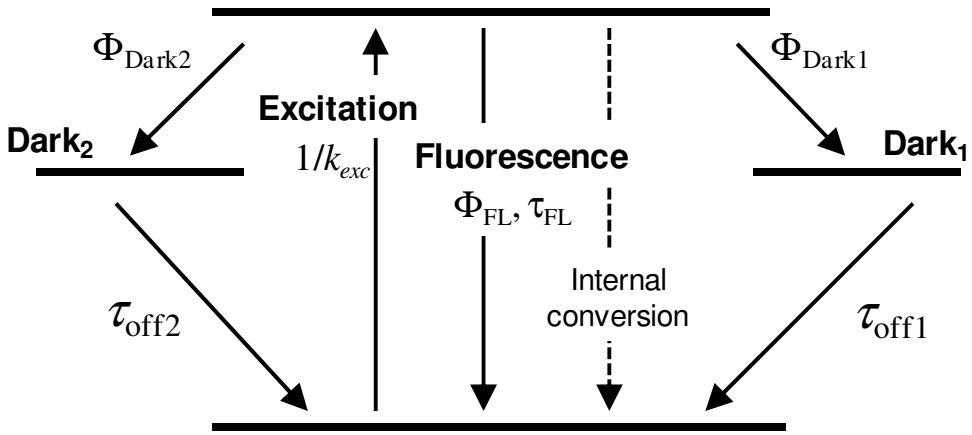

Figure 3.1 Jablonski diagram of 3-state model.

However, information about a dark state will be included in the photon waiting time which is collected after nonzero transitions to the dark state. Transitions to and from the dark state will increase the length of photon waiting time. Although the observation sequence for PbPHMM is photon waiting time, the hidden state is not an individual photophysical state. A new definition of hidden state is required to categorize various lengths of photon waiting time. I've defined a new set of states arising from photophysical transitions prior to photon detection. To avoid confusion, the word 'state' or 'hidden state' used without the prefix 'photophysical' is used to describe a PbPHMM state, not a photophysical state.

State 1 : A photon is detected without transition to any dark state.

State 2 : detected photon after transition to only the first dark state, $Dark_1$. Transitions to any other dark photophysical state are not included.

State 3 : detected photon after transition to the second dark state, $Dark_2$. Transition to $Dark_1$ is allowed. Transition to $Dark_3$ or higher order dark photophysical state is not included.

State *n* : detected photon after nonzero transition to $Dark_{n-1}$. Transition to $Dark_1$ ~ $Dark_{n-2}$ is allowed.

A schematic view of the photophysical processes before a state n photon is detected is shown in Figure 3.2.

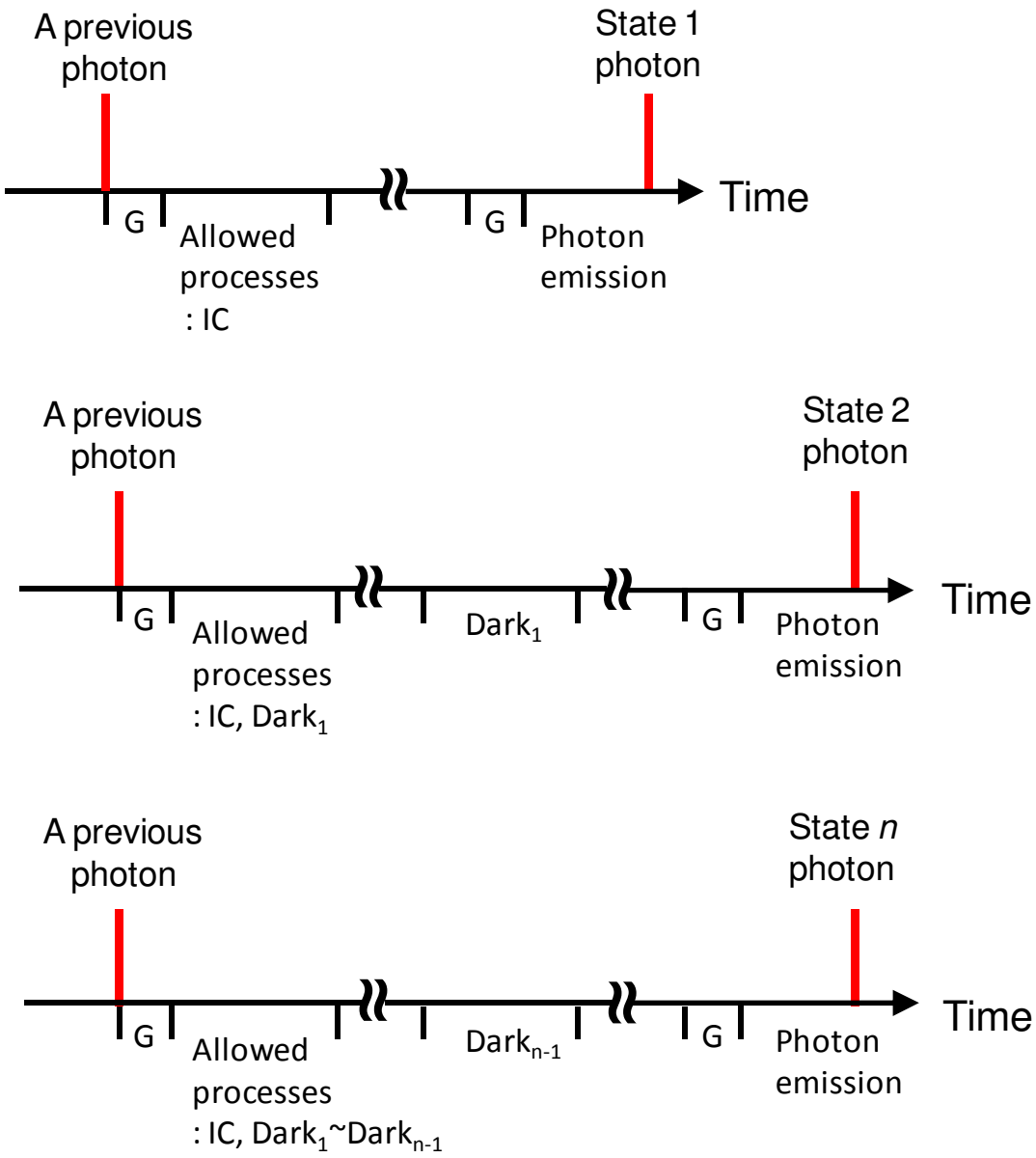

Figure 3.2 A schematic view of photophysical processes before a state *n* photon is detected. Ground photophysical state residence (G) is followed by any photophysical process. The waiting time of state 1 photon only includes the time information about singlet relaxation including internal conversion(IC) and fluorescence. By the definition of state 2, $Dark_1$ is accessed at least once before a state 2 photon is detected. Singlet relaxation and $Dark_1$ are allowed to be accessed multiple times, but other dark states are not allowed. In general, the waiting time of state *n* photon includes the information about singlet relaxation and $Dark_1$~$Dark_{n-1}$.

PbPHMM model building requires the existence of a relationship between photophysical parameters and PbPHMM model parameters. The photon waiting time at state 1, Δt_1, is the sum of duration for excitation to singlet state and relaxation to ground photophysical state without deviating to any other state. Its average value is $1/k_{exc}+\tau_{FL}$. The waiting time for both excitation and fluorescence lifetime follows exponential distributions. The probability density function of the sum of two continuous random variables is the convolution of those probability density functions [34]. Therefore, the probability density function of Δt_1 is the convolution of two exponential distributions. If the probability density function (PDF) of two random variables x and y are $f_x(x)$ and $f_y(y)$, then the PDF of a new random variable $z=x+y$ is as follows:

$$f_z(z) = f_x(x) * f_y(y) = \int_{-\infty}^{\infty} f_x(x) f_y(z-x) dx \tag{3.1}$$

The asterisk (*) in eq. 3.1 represents the convolution of two functions. Substituting mean values of excitation and fluorescence lifetimes, the PDF of Δt_1 is:

$$\begin{aligned} f_1(\Delta t_1) &= \int_{-\infty}^{\infty} f_x(x) f_y(\Delta t_1 - x) dx = \int_0^{\Delta t_1} k_{exc} e^{-k_{exc}x} e^{-x/\tau_{FL}} / \tau_{FL} dx \\ &= \frac{e^{-k_{exc}\Delta t_1} - e^{-\Delta t_1/\tau_{FL}}}{1/k_{exc} - \tau_{FL}} \end{aligned} \tag{3.2}$$

In state 2, singlet excitation is followed by triplet relaxation, re-excitation and singlet relaxation. Therefore, the photon waiting time at state 2, Δt_2, is the sum of the waiting times for excitation, the lifetime of $Dark_1$, excitation, and fluorescence. Its PDF is:

$$f_2(z) = f_{\text{excitation}} * f_{D_1} * f_{\text{internal conversion}} * f_{\text{fluorescence}} \tag{3.3}$$

$Dark_1$ can be accessed multiple times before a photon is detected. The population of transitions to $Dark_1$ is governed by a geometric distribution with probability Φ_{Dark1}. After entering and leaving $Dark_1$, the mean number of excitation-decay cycles is $1/\Phi_{FL}$ before a photon is detected. Within these transitions, the fraction of internal conversion and $Dark_1$ from photophysical state transitions before a photon detection are $\Phi_{IC}/(\Phi_{IC+}\Phi_{Dark1})$ and $[\Phi_{Dark1}/(\Phi_{IC+}\Phi_{Dark1})$, respectively. By the new definition of hidden states, $Dark_1$ is accessed at least once, and its mean population within one period of state 2 is $1+ \Phi_{Dark1}/ \Phi_{FL}$. Given equation 3.3, the PDF of photon waiting times at the *n*th state is:

$$f_n(z) = f_{\text{excitation}} * f_{D_1} * f_{D_2} \cdots * * f_{D_{n-1}} * f_{\text{internal conversion}} * f_{\text{fluorescence}} \tag{3.4}$$

Transition probabilities are related to the quantum yields of the photophysical states. With sufficient collecting time of photons, the population ratio of photophysical states is completely determined by the ratio of quantum yields. In photophysical state *n*, $Dark_{n-1}$ is always accessed at least once, and the transition to $Dark_1$ ~ $Dark_{n-2}$ is allowed. In the photophysical 4-state model, for example, $Dark_1$ may be involved before a photon at state 3 is detected, but $Dark_3$ is not allowed. Transition probabilities of each hidden state in 3-state model, (not photophysical states), are as follows:

$$\begin{aligned}
p(state\,1 \mid state\,x) &= p_{x1} = \frac{\Phi_{FL}}{\Phi_{FL} + \Phi_{Dark1} + \Phi_{Dark2}} \\
p(state\,2 \mid state\,x) &= p_{x2} = \frac{\Phi_{FL}\Phi_{Dark1}}{(\Phi_{FL} + \Phi_{Dark2})(\Phi_{FL} + \Phi_{Dark1} + \Phi_{Dark2})} \\
p(state\,3 \mid state\,x) &= p_{x3} = \frac{\Phi_{Dark2}}{\Phi_{FL} + \Phi_{Dark2}}
\end{aligned} \tag{3.5}$$

3.2.2 Effect of non-unity detection efficiency

Non-unity detection efficiencies lead to uncollected photons. In this case, not every fluorescent singlet relaxation is observed. Φ_{Det} is the probability that an emitted photon is detected. Therefore, eq. (3.4) and (3.5) are modified to analyze realistic photon time traces.

Photophysical processes a within photon waiting time at state 1 are as follows. As mentioned in the previous section, a photon at state 1 is detected without transition to any photophysical dark state. Before the photon is detected, additional photons at state 1 are missed due to non-unity detection efficiency. Additional singlet relaxations are not directly recorded due to nonradiative internal conversion. These processes correspond to a series of singlet transitions with photon misses followed by single photon detection. In order to describe the population of missed singlet relaxation and the distribution of photon waiting time mathematically, the distribution of waiting times is written as a geometric distribution.

The geometric distribution describes the probability that x trials will be needed before an event occurs [35]. The number of trials is called a geometric random number and is a non-negative integer. If the probability that an event A occurs $P(A)=p$, the probability mass function (PMF) $f(x)$, mean $E(x)$, and variance $V(x)$ of geometric random number x are as follows:

$f(x)=p(1-p)^x$, when $P(A)=p$ and $x=0,1,2,\ldots$

$$E(x)=\frac{1-p}{p}, V(x)=\frac{1-p}{p^2} \tag{3.6}$$

The PMF for discrete random variables corresponds to the probability distribution function for continuous random variables.

Describing waiting times, it's more convenient to consider the probability of allowed processes. If q is the probability of allowed processes before the event A occurs, eq. (3.6) can be changed as follows:

$$f(x) = q^x(1-q),\ x = 0,1,2,\cdots$$

$$E(x) = \frac{q}{1-q},\ V(x) = \frac{q}{(1-q)^2}, \tag{3.7}$$

where q is the sum of the probability of allowed processes.

The photon waiting time at photon state 1 is the sum of the dwell time in the ground photophysical state, the lifetime of internal conversion, and the fluorescence decay time for all missed photons and the detected photon. Therefore, ground photophysical state residence, internal conversion, and missed photon emission are the allowed photophysical processes. Since time between photon detection events is the observable, ground photophysical state residency is always followed by singlet transition, and the sum of probability of allowed states is given by the following equation:

p = P(missing emitted photons) + P(internal conversion)

$$= (1-\Phi_{\text{Det}})\Phi_{\text{FL}} + \Phi_{\text{IC}} \tag{3.8}$$

Therefore, the PMF of the number of allowed processes, singlet relaxations, before state 1 photon is detected and its average are as follows:

$n_1^{singlet}$: The number of singlet transition before photon detection at state 1

$$\text{PMF}\left(n_1^{singlet} = x\right) = (1-p)\,p^x,\ p = (1-\Phi_{\text{Det}})\Phi_{\text{Fl}} - \Phi_{\text{IC}}$$

$$E\left(n_1^{singlet}\right) = \frac{(1-\Phi_{\text{Det}})\Phi_{\text{Fl}} - \Phi_{\text{IC}}}{1-(1-\Phi_{\text{Det}})\Phi_{\text{Fl}} + \Phi_{\text{IC}}} \tag{3.9}$$

Dwell time at the ground photophysical state is exponentially distributed, and its mean is $1/k_{exc}$. k_{exc} is the excitation rate of a molecule when it is illuminated by light with the intensity I_{prim}.

$$k_{\text{exc}} = \frac{\sigma_{abs} I_{\text{prim}} \lambda_{\text{prim}}}{hc\left(1 + \frac{I_{\text{prim}}}{I_{\text{sat}}}\right)},\ I_{\text{sat}} = \frac{hc}{\sigma_{abs} \lambda_{\text{prim}} \tau_{\text{Fl}}}, \tag{3.10}$$

where σ_{abs} is the absorption cross section, I_{prim} the primary excitation intensity, λ_{prim} the wavelength of primary excitation, h the Planck constant, c the speed of light, I_{sat} the saturation intensity, and τ_{FL} the fluorescence lifetime. Consequently, the photon waiting time at state 1 is the product of $n_1^{singlet}$ and $(\Phi_{FL}\tau_{FL} + \Phi_{IC}\tau_{IC})/(\Phi_{FL}+\Phi_{IC})$.

The probability density function $f(x)$ of geometrically-populated sum of exponential random number is as follows:

$$\begin{aligned} f(x)dx &= p & x &= 0 \\ f(x)dx &= (1-p)\frac{Exp(-x/(\tau/p))}{\tau/p} & x &> 0 \end{aligned} \tag{3.11}$$

where p is the probability of event occurring, and τ is the mean exponential random number. By replacing p and τ by 1-p in eq (3.12) and $1/k_{exc}$ respectively, the sum of dwell time at the ground photophysical state Δt_{Ground} is distributed by the following equation.

$$f(x)dx = \frac{Exp\left[-x\,k_{\text{exc}}\left(1-\Phi_{\text{Fl}}\Phi_{\text{Det}}-\sum\Phi_{\text{D}}\right)\right]}{1/k_{\text{exc}}\left(1-\Phi_{\text{Fl}}\Phi_{\text{Det}}-\sum\Phi_{\text{D}}\right)},\ x = \Delta t_{\text{Ground}} \tag{3.12}$$

Convolving the sum of fluorescence lifetime, the resulting probability distribution function (PDF) of photon waiting time at state 1 is

$$f(x)dx = \frac{Exp(-x/t_1) - Exp(-x/t_2)}{t_1 - t_2} \tag{3.13}$$

where $t_1=1/[k_{exc}(1-\Phi_{Det}\Phi_{FL}-\Sigma\Phi_{Dark})]$ and $t_2=-\tau_{FL}(1-\Phi_{Det}\Phi_{FL}-\Sigma\Phi_{Dark})$.

Observed photon waiting times for state 2 are affected by the detection efficiency due to the possibility that all photons during the on-period are not detected. The conditional probability of missing photons at a given on-state is:

$$P(\text{no photon} \mid \text{on}) = \frac{(1-\Phi_{Det})\sum\Phi_{ISC}}{(1-\Phi_{Det})\sum\Phi_{ISC}+\Phi_{Det}} \tag{3.14}$$

Photophysical processes in state 2 are divided into two stages, which are accessed until transition to $Dark_1$ and photophysical states accessed until photon detection after the first transition to $Dark_1$. In the first stage, the allowed photophysical processes are fluorescence decay without photon detection and internal conversion. In the second stage, transition to $Dark_1$ is allowed as well as internal conversion and fluorescence decay. The average population of photophysical states at state 2 is:

Stage 1

$$\text{average number of transitions to } s = \left(\frac{1}{1-p_1}-1\right)\frac{\Phi_s}{p_1}$$

$$p_1 = \Phi_{\mathrm{IC}} + \Phi_{\mathrm{FL}}(1-\Phi_{\mathrm{Det}})$$

Stage 2

$$\text{average number of transitions to } s = \left(\frac{1}{1-p_2}-1\right)\frac{\Phi_s}{p_2}$$

$$p_2 = \Phi_{\mathrm{IC}} + \Phi_{\mathrm{FL}}(1-\Phi_{\mathrm{Det}}) + \Phi_{\mathrm{Dark1}} \tag{3.15}$$

in which Φ_s represents the quantum yield to generate allowed photophysical state s including Φ_{IC} and $\Phi_{FL}(1-\Phi_{Det})$ for stage 1, and Φ_{IC}, $\Phi_{FL}(1-\Phi_{Det})$, and Φ_{Dark1} for stage 2. Sums of the allowed photophysical processes within stage 1 and stage 2 are denoted by p_1 and p_2 respectively where $p_1=\Phi_{IC}+\Phi_{FL}(1-\Phi_{Det})$ and $p_2=\Phi_{IC}+\Phi_{FL}(1-\Phi_{Det})+\Phi_{Dark1}$.

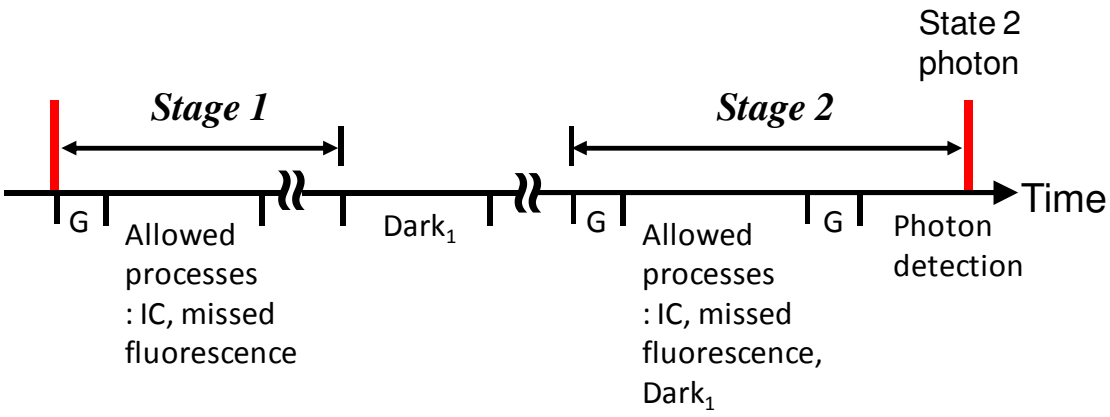

Figure 3.3 The allowed photophysical processes before a state 2 photon is detected when Φ_{Det} is non-unity. $Dark_1$ is accessed at least once, so it is convenient to consider the convoluted waiting times into two stages.

A schematic view of photophysical processes before a state 2 photon is detected with non-unity detection efficiency is shown in Figure 3.3.

The PDFs of dwell time at each photophysical state are product distributions of geometric and exponential distributions as shown in state 1. Consequently the PDF of photon waiting times at state 2 is the convolution of photophysical state dwell time PDFs. The duration at stage 1 at state 2, τ(stage 1, state 2), is the sum of dwell time of internal conversion, missed fluorescence decay, and one period of decay via $Dark_1$ transition. The PDF of the duration at the stage 1 at state 2 is not a simple convolution due to the nonzero probability of zero internal conversion and missed fluorescence decay.

$$f[\tau(\text{stage}\,1,\text{state}\,2)]d\tau = (1-p_1)f_{\text{ground,part1}} * f_{\text{Dark1,part1}} + p_1 f_{\text{ground,part1}} * f_{\text{IC, part1}} *$$

$$f_{\text{FL,no photon detection part1}} * f_{\text{ground}} * f_{\text{Dark1}} \tag{3.16}$$

In the same way, the PDF of the duration of the stage 2 at state 2, τ(stage 1, state 2), is:

$$f[\tau(\text{stage}\,2,\text{state}\,2)]d\tau = (1-p_2)f_{\text{ground}} * f_{\text{FL}} + p_2 f_{\text{ground,part2}} * f_{\text{Dark1,part2}} * f_{\text{IC, part1}} *$$

$$f_{\text{FL,no photon detection, part1}} * f_{\text{ground}} * f_{\text{FL,photon detection}} \tag{3.17}$$

Replacing all symbols by photophysical parameters, the photon waiting time PDFs, τ(stage 1, state 2) and τ(stage 2, state 2), are:

$$f_{\text{phase 1, state2}}(\tau)d\tau = (1-p_1)\,ED[1/k_{\text{exc}}] * ED[\tau_{\text{off1}}] + p_1\left\{ED\left[(1/k_{\text{exc}}+\tau_{\text{IC}})\frac{\Phi_{\text{IC}}}{p_1(1-p_1)}+\right.\right.$$

$$\left.\left.(1/k_{\text{exc}}+\tau_{\text{FL}})\frac{\Phi_{\text{FL}}(1-\Phi_{\text{Det}})}{p_1(1-p_1)}\right] * ED[1/k_{\text{exc}}] * ED\left[\frac{\tau_{\text{Fl}}}{p_1}\right]\right\} \tag{3.18}$$

$$f_{\text{phase 2,state2}}(\tau)d\tau = (1-p_2)\,ED[1/k_{\text{exc}}] * ED[\tau_{\text{FL}}] +$$

$$p_2\left\{ED\left[\left(1/k_{\text{exc}}+\tau_{\text{IC}}\right)\frac{\Phi_{\text{IC}}}{p_2\left(1-p_2\right)}+\left(1/k_{\text{exc}}+\tau_{\text{FL}}\right)\frac{\Phi_{\text{FL}}\left(1-\Phi_{\text{Det}}\right)}{p_2\left(1-p_2\right)}+\right.\right.$$

$$\left.\left.\left(1/k_{\text{exc}}+\tau_{\text{off1}}\right)\frac{\Phi_{\text{Dark1}}}{p_2\left(1-p_2\right)}\right]*ED\left[1/k_{\text{exc}}\right]*ED\left[\tau_{\text{Fl}}\right]\right\} \tag{3.19}$$

where p_1= Φ_{IC}+Φ_{FL}(1- Φ_{Det}), and p_2=Φ_{IC}+Φ_{FL}(1- Φ_{Det})+Φ_{Dark1}, and $ED[x_0] = e^{-x/x_0}/x_0$. The PDF of photon waiting times at state 2 is the convolution of eq. (3.18) and eq. (3.19). These equations can be applied to derive PDFs of photon waiting times at other states by dividing photophysical processes occurring between any two photon arrivals into two stages and estimating dwell time distributions of allowed processes.

Transition probabilities with non-unity detection efficiency are derived from the populations of photophysical processes that determine hidden state of detected photons. By the definition of hidden states in section 3.2.1, Dark_2 is included only within photon waiting time at state 3. With sufficient numbers of excitation-deexcitation cycles, population ratios of photophysical processes are proportional to their quantum yield. Therefore, the population ratio of state 3 is Φ_{Dark2} divided by the mean accessed number of times Dark_2 is accessed within photon waiting time at state 3. Transition probabilities with non-unity detection efficiency are as follows (see Appendix B):

Transition probabilities in 3-state model with non-unity Φ_{Det}

$$\begin{aligned}
p\left(state\ 1 \mid state\ x\right) &= p_{x1} = \frac{\Phi_{\text{Det}}\Phi_{\text{FL}}}{\Phi_{\text{Det}}\Phi_{\text{FL}}+\Phi_{\text{Dark1}}+\Phi_{\text{Dark2}}} \\
p\left(state\ 2 \mid state\ x\right) &= p_{x2} = \frac{\Phi_{\text{Det}}\Phi_{\text{FL}}\Phi_{\text{Dark1}}}{\left(\Phi_{\text{Det}}\Phi_{\text{FL}}+\Phi_{\text{Dark2}}\right)\left(\Phi_{\text{Det}}\Phi_{\text{FL}}+\Phi_{\text{Dark1}}+\Phi_{\text{Dark2}}\right)} \\
p\left(state\ 3 \mid state\ x\right) &= p_{x3} = \frac{\Phi_{\text{Dark2}}}{\Phi_{\text{Det}}\Phi_{\text{FL}}+\Phi_{\text{Dark2}}}
\end{aligned} \tag{3.20}$$

3.2.3 Background noise

The collection of experimental data is subject to many types of undesired fluctuations in signal, as well as noise. Electronic instruments often generate noise, such as Johnson noise and shot noise [36]. Mechanical vibration from optical setup and background signal also contribute noise. Mechanical noise can be reduced by performing experiments on a vibration-damping optical table. The routing module for multiple detectors picks up inductive and capacitive noise, and radio frequency noise from the environment such as laser system, which can be reduced by proper shielding [37]. Increased noise level by the number of detectors should be considered since multiple APD were used in the setup.

Most of the noise processes can be assumed to be Poisson distributed. Noise pulse arrival can be considered as a renewal process since waiting times of Poisson distributed noise pulses are exponentially-distributed random numbers. The superposed process of independent Poisson processes is, itself, a Poisson process with the rate of the respective sum of individual distribution waiting times [38]. Therefore, noise included in collected data can be considered as the combination of Poisson noise from multiple sources. The photon waiting times from a single emitter, however, are not exponential random numbers, which follow the convolution of multiple exponential distributions. In this case the contaminated PDF by Poisson noise is not a single exponential distribution.

The new PDF can be derived through reliability theory [39]. Suppose $F(x)$ denotes the probability that a system operates until time x, and $f(x)$ is the probability density function of a time to failure x. The failure rate function $\nu(x)$ represents the conditional probability per unit time that a system operating at time x fails between x and $x+dx$.[40-

43]. The recurrence rate function is the inverse of the forward recurrence time that is the average waiting time from x until failure if the system operates until time x [44]. Suppose the PDF of waiting time x for failure of renewal process is $f(x)$, then the failure rate function $\nu(x)$ and the recurrence rate function $\eta(x)$ are

$$\nu(x) = \frac{f(x)}{\int_x^\infty f(t)dt}$$
$$\eta(x) = \frac{1-F(x)}{\int_x^\infty (1-F(t))dt}, \quad \text{where } F(x) = \int_0^x f(t)\,dt \tag{3.21}$$

When the PDFs of each renewal processes are given, the failure rate function of the process in which renewal processes are superposed is given in eq. (3.22)

$$\nu_s(x) = \frac{\sum_{i=1}^n \eta_i(x) \cdot \left(\nu_i(x) + \sum_{j=1, j\neq i}^n \eta_j(x)\right)}{\sum_{i=1}^n \eta_i(x)} \tag{3.22}$$

Using eq. (3.21) and (3.22), the PDF of waiting time of a superposed process can be derived. For example, if the waiting times of a renewal process are distributed by the convolution of two exponential distributions with respective means τ_1 and τ_2, then the resulting PDF of detection contaminated with Poisson noise, $f_s(x)$, will be:

Simulated PDF with Poisson and convolved random numbers was compared with PDF in eq. (3.23) in Figure 3.2.

$$f_s(x) = \frac{e^{-\lambda x}}{(\tau_1+\tau_2+1/\lambda)/\lambda}\left(\frac{e^{-x/\tau_1}(\tau_1+1/\lambda)^2}{(\tau_1-\tau_2)} + \frac{e^{-x/\tau_2}(\tau_2+1/\lambda)^2}{(\tau_2-\tau_1)}\right) \tag{3.23}$$

The PDF of the superposition of signal from convolution of n different exponential distributions and Poisson noise with an average count rate λ is shown in Eq. (3.24).

$$f_s(x) = \frac{e^{-\lambda x}}{(\sum_{i=1}^n \tau_i + 1/\lambda)/\lambda} \sum_{i=1}^n \frac{e^{-x/\tau_i}\tau_i^{n-2}(\tau_i+1/\lambda)^2}{\prod_{j,j\neq i}^n (\tau_i-\tau_j)} \tag{3.24}$$

Background noise causes deviation in the transition matrix as well. The deviation is due to the definition of the state assignment of detected signal. For example, with sufficient noise level, several signal pulses may be collected within the waiting time at state 2. Diagonal elements of transition matrix for hidden states represent a probability of staying the previous state. This value increases by the amount related the number of detected signal pulses. The transition probability staying at state 2 with nonzero background noise, a'_{ij}, as follows.

$$a'_{22} = \frac{d_2 - 1}{d_2} = \frac{\Delta t_2 \times \text{noise level}}{\Delta t_2 \times \text{noise level} + 1},$$

$$\Delta t_2 = \left[\frac{1}{\Phi_{\text{Det}}\Phi_{\text{FL}} + \Phi_{\text{D1}} + \Phi_{\text{D2}}} + \frac{1}{\Phi_{\text{Det}}\Phi_{\text{FL}} + \Phi_{\text{D2}}} \right] \left[1/k_{exc} + \tau_{\text{IC}}\Phi_{\text{IC}} + \tau_{\text{FL}}\Phi_{\text{FL}}(1-\Phi_{\text{Det}}) \right] + \tau_{\text{D1}} \left(1 + \frac{\Phi_{\text{D1}}}{\Phi_{\text{FL}}\Phi_{\text{Det}} + \Phi_{\text{D2}}} \right) \quad (3.25)$$

where d_2 is the duration of adjacent state 2 and Δt_2 is the mean photon waiting time at state 2. Each row of transition matrix needs to be normalized after its diagonal elements are modified by background noise.

3.2.4 Multiple observations

Typically the accuracy of estimated parameters by HMM is improved with the increased number of data points. When the number of observations is limited, analyzing multiple observation sequences can improve fitting performance [45, 46]. When K observation sequences, $\boldsymbol{x}=(x^{(1)}, x^{(2)},\ldots, x^{(K)})$ are available, the estimated transition $\bar{a}_{ij}$ and emission probabilities $\bar{b}_i(l)$ are given by

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{c_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(x_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{c_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_{t+1}^k(j)}, \quad \bar{b}_i(l) = \frac{\sum_{k=1}^{K} \frac{1}{c_k} \sum_{t=1, x_t=l}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{c_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{3.26}$$

where c_k is a weighting factor for each sequence [45, 46]. Various kinds of weighting factors were tested including unit weights [13, 47] and average likelihood of all models over all observation sequences [45] or a specific percentile of the sequences chosen by their likelihood (called Windsorised method) [47]. Applying HMM to multiple observation sequences is specifically useful for data analysis of single molecule spectroscopy as typical single molecule experimental data are very short due to photobleaching and noise. Increased number of short datasets can improve fitting performance by eq. (3.26) and the unit weight was used in this study [13, 47].

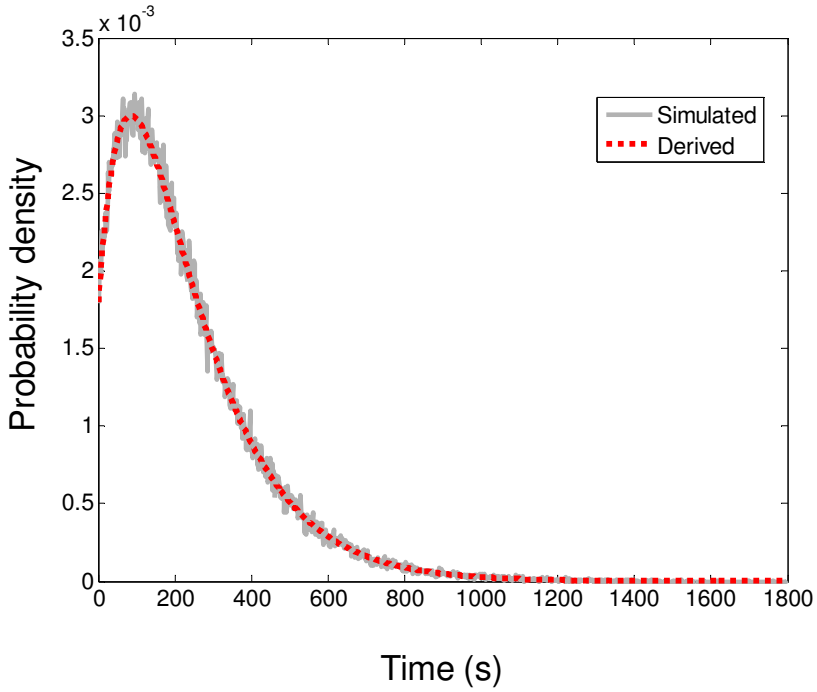

Figure 3.4 Derived and simulated PDF of superposed renewal process. PDF of waiting time of renewal process was the convolution of two exponential distributions with the mean τ_1=100 and τ_2=200 respectively. Mean waiting time of Poisson distributed noise was 1000. Simulated PDF (gray solid) was presented by generating convoluted and Poisson distributed random numbers and calculating histogram of superposed waiting times. Derived PDF (red dotted) was from eq. (3.23).

3.3 Methods

Using simulated fluorescence photon trajectories from HMM-based generation, fitting performance of PbPHMM was evaluated under diverse conditions. Three-state model and photophysical parameter values are presented in Figure 3.1 and Table 3.1, respectively.

The algorithm for generating photon time trajectories is shown in Figure 3.3, which is based on hidden Markov chain generating algorithm called Gillespie algorithm [48, 49]. First, a Markov chain of photophysical processes is generated according to the transition probabilities between photophysical processes. The allowed photophysical processes in 3-state model are the ground photophysical state, nonradiative internal conversion, missed fluorescence, fluorescence photon detection, and two dark states ($Dark_1$ and $Dark_2$). The 6×6 transition matrix P for the Markov chains of the sequence of photophysical processes is:

$$P = \{p_{ij}\} = \begin{bmatrix} 0 & \Phi_{IC} & \Phi_{FL}(1-\Phi_{Det}) & \Phi_{FL}\Phi_{Det} & \Phi_{Dark1} & \Phi_{Dark2} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.27)$$

Ground photophysical state, internal conversion, the missing of emitted photon, photon detection, $Dark_1$, and $Dark_2$ were assigned to the state 1 to state 6 of the Markov chain respectively. Zero probability in p_{11} means that any excitation and decay occurs after ground photophysical state residence. Unity probabilities in p_{21}~ p_{61} and zero probabilities in p_{22}~ p_{66} indicate that any photophysical process is followed by ground photophysical state residence.

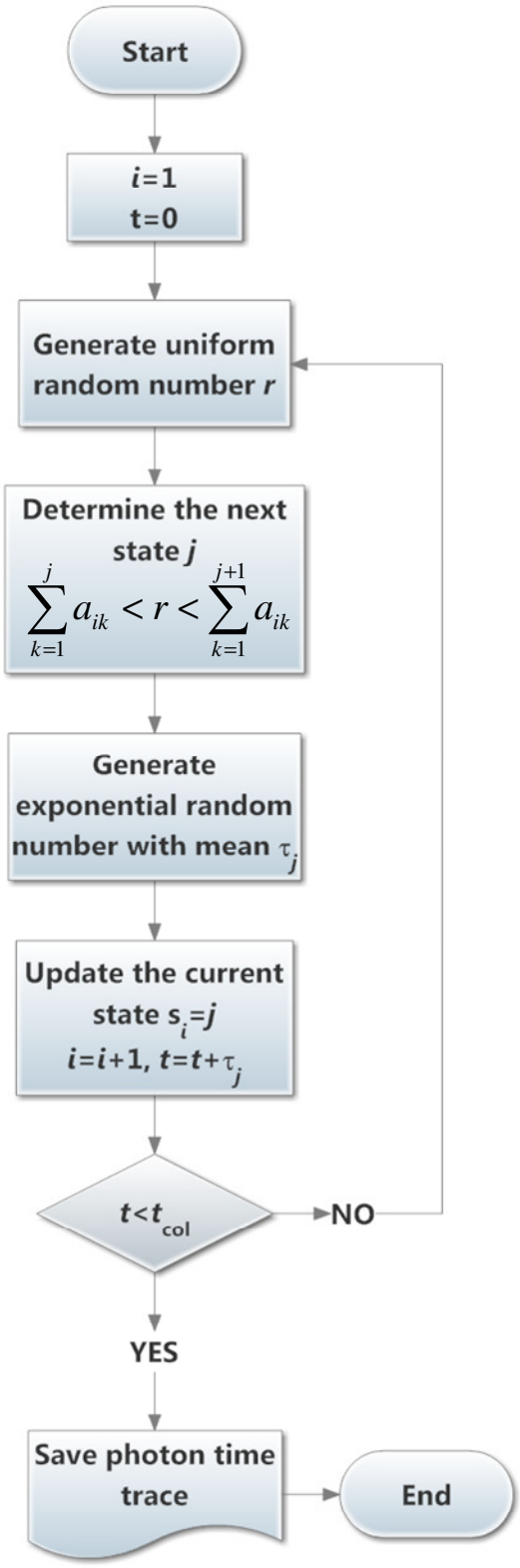

Figure 3.5 A photon generation algorithm based on HMM for 3-state model with two dark states.

Table 3.1 Photophysical parameters used in generating photon trajectories.

	Photophysical parameter	Symbol	Value
Varying parameters	Excitation intensity	I_{prim} ($W\ cm^{-2}$)	250~1,250
	Quantum yield ($Dark_1$)	Φ_{Dark1}	0.01~0.09
	Lifetime ($Dark_1$)	τ_{off1} (s)	2×10^{-5} ~ 1×10^{-3}
	Detection efficiency	Φ_{Det}	0.05~1
	Noise level		0~5000 s^{-1}
Fixed parameters	Extinction coefficient	ε ($M^{-1}\ cm^{-1}$)	100,000
	Fluorescence lifetime	τ_{FL} (s)	3.5×10^{-9}
	Fluorescence quantum yield	Φ_{FL}	0.5
	Quantum yield ($Dark_2$)	Φ_{Dark2}	0.001
	Lifetime ($Dark_2$)	τ_{off2} (s)	1×10^{-3}
	Time step	t_{step} (s)	1×10^{-7}

After the sequence of photophysical processes is generated, a sequence of exponential random numbers is generated according to the average lifetime of each photophysical processes. The cumulative sum of exponential random numbers is the time stamp of photophysical processes. Since the detected photons are the only available observables, the time stamp of photon detection is saved as photon arrival times. Exponential random numbers for noise are generated and mixed into photon trajectories. Photon arrival times were rounded to 100 ns to mimic typical time traces collected with time-correlated single photon counting (TCSPC) setup using an avalanche photodiode.

The values of photophysical parameters were adjusted for generating realistic photon traces from existing fluorescent probes. The number of excitation/de-excitation cycles was fixed at 1×10^6, which is the typical limit of photostability of various organic dyes and fluorescent proteins in single molecule experiments [50-52]. The photon detection efficiency was fixed at 5% for all simulations except when the effect of detection efficiency on fitting performances was studied. 50 time traces were generated with given different values of experimental and photophysical parameters, which include dark state quantum yield, dark state lifetime, excitation intensity, and noise level.

The algorithm of PbPHMM is shown in Figure 3.4. The analysis starts from initial photophysical parameters determined by autocorrelation function fitting based on an *n*-state model. Photophysically-relevant transition and emission matrices were estimated using initial parameters. These matrices were trained by the Baum-Welch algorithm as shown eq. (1.12) and (1.13) using initial model parameters. After every iteration of training, photophysical parameters were extracted from trained model parameters by solving eq. (3.13), (3.18), (3.19), (3.20), (3.24), and (3.25) simultaneously, which are

then used in the next iteration. Photophysically-constrained Baum-Welch training was repeated until log-likelihood was maximized. BICs were calculated with trained transition and emission matrices using eq. (2-1). The degrees of freedom of trained model parameters d in eq. (2-1) is S^2+2S-3 where S is the number of hidden states. The transition matrix is an $S \times S$ matrix, and each element is the transition probability to be trained. One restriction per each row exists that the sum of each row should be one, which makes the degree-of-freedom of the transition matrix $S(S-1)$. The emission matrix of state 1 is the convolution of exponential distributions, and their average τ_{IC}, τ_{FL}, and $1/k_{exc}$ are all known parameters. The emission matrix of state n includes the information of $Dark_1$~$Dark_{n-1}$ of which parameters are the quantum yields Φ_{Dark1}~ $\Phi_{Dark\ n-1}$ and lifetimes τ_{off1}~ $\tau_{off\ n-1}$. The contribution of emission matrices to d is, therefore, $2(S-1)$. The initiation matrix is a column vector of which elements are the probability of S different states at the beginning with one restriction. Therefore, the total degree-of-freedom d is $S(S-1) + 2(S-1) + S-1=S^2+2S-3$.

The same calculations were repeated after adding or removing another dark state. As mentioned in Chapter 2, the number of hidden states was determined at the largest BIC.

Two conditions should be met for statistical relevance during PbPHMM training. First, populations of all hidden states must be positive integers. After model parameters were trained at a given number of hidden states, the state population was calculated from the reconstructed state sequence by the Viterbi algorithm. If one of the hidden states was not accessed in the fitting at n states, the n-1 state model is automatically selected. The second condition is that all different states should be statistically distinguishable from

each other. In place of a localization error test in the time-binned analysis, a Wilcoxon rank-sum test was used to check the existence of unnecessary (overfitted) states. The rank-sum test is a kind of nonparametric test and efficient for heavily-tailed distributed samples [35]. In this study, confidence level was set to 95% to test overlap between photons from each state.

Two evaluation categories of the fitting performance of PbPHMM at different conditions are a relative error of estimated photophysical parameters and model selection accuracy. Relative errors of estimated parameters were calculated as percentages compared to real parameters used in generating photons.

$$\text{Relative error} = \frac{x_{\text{estimated}} - x_{\text{real}}}{x_{\text{real}}} \times 100 \tag{3.28}$$

The model selection accuracy was the probability that the resulting number of dark states is the same as one in the photophysical model for generating photons.

To compare parameter estimation of single data and multiple observation sequences, the average and standard deviation of relative errors of photophysical parameters estimated from 50 individual traces with 30,000 photons were compared to those relative errors from photophysical parameters re-estimated over 50 time traces. Relative errors of photophysical parameters from multiple observations were calculated from re-estimated transition and emission matrices using eq. (3.26) with unit weighting factors. All codes for photon generation and PbPHMM analysis were written in Matlab software (R2013b, Mathworks), and all calculations were performed on Georgia Tech PACE-managed computer clusters.

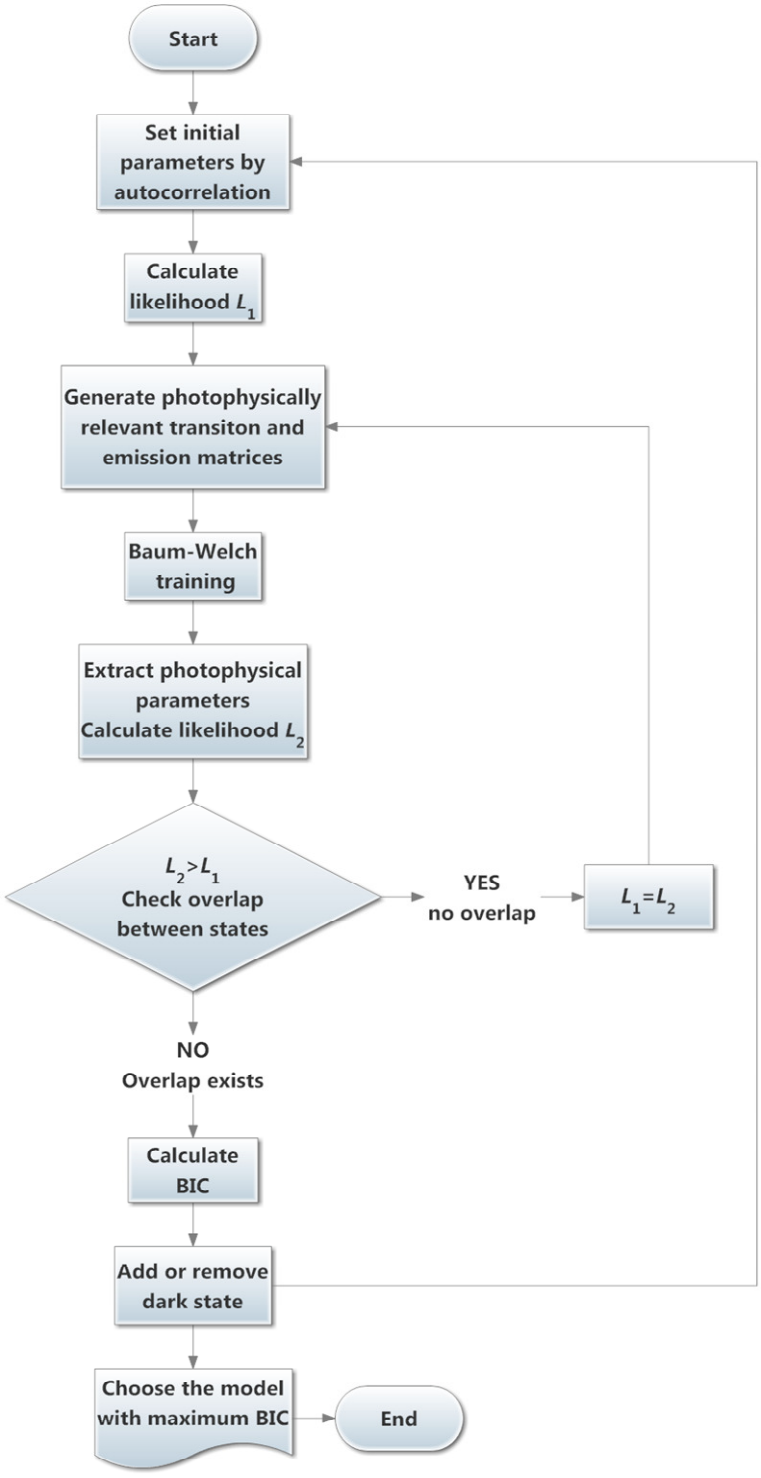

Figure 3.6 A PbPHMM algorithm.

3.4 Results and discussion

3.4.1 Photon generation by HMM-based algorithm

All photons for testing fitting performance of PbPHMM were generated with an HMM-based algorithm. First the sequence of photophysical processes is generated according to the quantum yields and lifetimes of those processes. Then, the time stamps of photon detection are recorded. The example of generated photon trace and underlying photophysical processes are presented in Figure 3.5. The positions of vertical lines in Figure 3.5 (a) represent photon arrival times. The stair step plot in Figure 3.5 (b) depicts transitions of photophysical processes. Y-values of the plot represent singlet relaxation, decay through $Dark_2$, $Dark_1$, and ground photophysical states, and x-values represent elapsed time after decay also known as dwell time. The singlet decay looks like a pulse because the lifetime of singlet decay is short (3.5 ns) compared to the dwell time at ground photophysical state or dark state lifetime (~μs).

The position of most of photons in Figure 3.5(a) exactly matches those of singlet decay in Figure 3.5(b). However photon emitted from singlet decay in dashed circles in Figure 3.5(b) were not detected due to non-unity detection efficiency. Moreover a photon may not be emitted during singlet decay due to non-unity fluorescence quantum yield. A photon with a black arrow in Figure 3.5(a) was detected during dark state relaxation, which means it is a background signal. The hidden states of photons were defined as mentioned in Section 3.2.1.

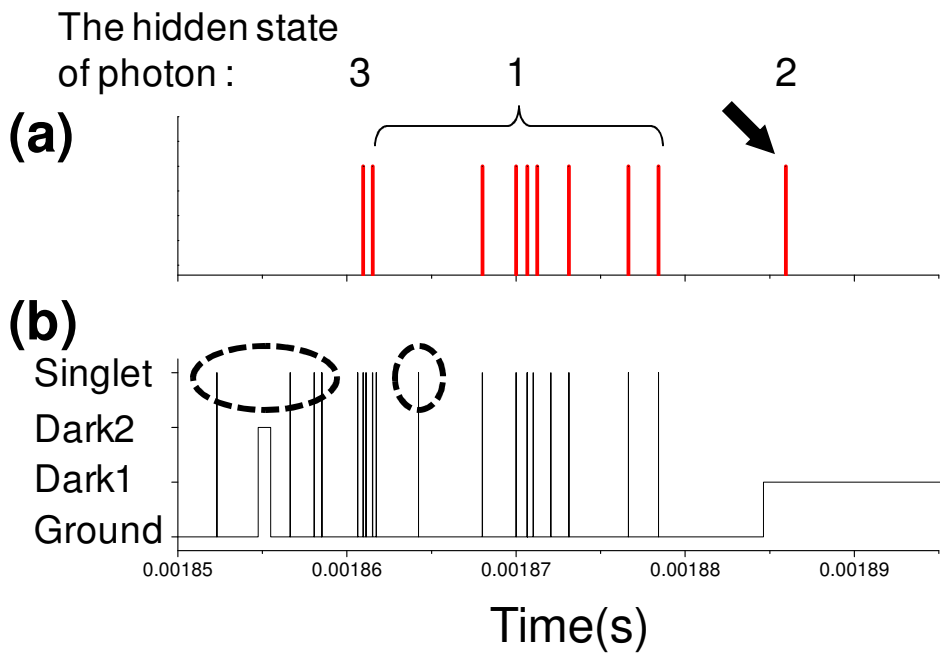

Figure 3.7 Detected photons and underlying photophysical processes generated by HMM-based algorithm. (a) The horizontal position of vertical lines depicts photon arrival time. The numbers above photons show the hidden state of photons. A black arrow points a background photon. (b) Dwell time at each photophysical processes. Y-values from top to bottom represents singlet relaxation, decay via $Dark_2$ and $Dark_1$, and ground photophysical state.

3.4.2 Comparison of PbPHMM to BW algorithm

PbPHMM analysis for photophysical modeling was compared to traditional BW algorithm in robustness and fitting performance. Time trace of 30,000 photons was generated with Φ_{Det}=0.05, I_{prim}=300 W cm^{-2}, and background count=1000 s^{-1}. Other parameters are shown in Table 3.1. Photophysical parameters including dark state quantum yield and lifetime were estimated by analyzing the time trace with BW and PbPHMM algorithm with 3-state model. Initial parameters were set randomly, and training continued until the log-likelihood was maximized. The Log-likelihood of photons and relative errors of photophysical parameters are shown in Figure 3.6(a) and (b) respectively. The thick dot line in Figure 3.6(a) is the log-likelihood calculated during iteration by BW algorithm, and the red solid line is from PbPHMM. The Baum-Welch (BW) algorithm did not maximize log-likelihood even after 200 iterations (not displayed in Figure 3.6(a)), the estimated log-likelihood became more positive than the value from real parameters in dash-dot lines.

In general larger likelihood means a better fitting. However, the maximization step in the BW algorithm is basically a maximum likelihood estimation, which is susceptible to overfitting in parameter estimation, especially for short data [53, 54]. The problem of overfitting can be reduced by incorporating an appropriate prior distribution of model parameters $P(\lambda)$[55-58]. Without prior knowledge about model parameters, a uniform distribution can be used as the prior probability distribution [59].

In PbPHMM, however, model parameters are not uniformly distributed due to photophysical constraints. Transition and emission matrices are functions of photophysical parameters as shown in Section 3.2, and the range of photophysical

parameters are limited: the dark state lifetime should be positive numbers, and the sum of fluorescence and dark state quantum yields should be a positive number not larger than one. With those constraints, log-likelihood from PbPHMM converged faster than the BW algorithm to the real value with small relative error (<0.001%). Relative errors of photophysical parameters calculated by two algorithms were tracked during iterations in Figure 3.6(b).

The relative errors from PbPHMM converged faster to smaller values (<10%) while Baum-Welch algorithm showed larger errors, especially of Φ_{Dark2}. The reason for large errors is that the BW algorithm may return photophysically-irrelevant transition and emission matrices after each iteration due to uniform prior distributions. These matrices induce large errors in estimating photophysical parameters even if they cause better likelihood. Figure 3.6(c) shows that the emission matrix from PbPHMM overlaps with one from real parameters very well, but the BW algorithm emission matrix significantly deviates from the real emission matrix.

PbPHMM estimated the parameters accurately even with random initial conditions as displayed in Figure 3.6(d) and (e). Robustness against deviating initial conditions was tested by comparing relative errors of photophysical parameters estimated using BW algorithm and PbPHMM. For convenience, root mean squared sum of relative errors were presented instead of individual error values. PbPHMM resulted in relative errors less than 5% even if the initial parameters were far from the real values, while the errors from BW algorithm scattered over 40 to 60%. The number of hidden states was determined correctly at maximized BIC by PbPHMM training as shown in Figure 3.6(f).

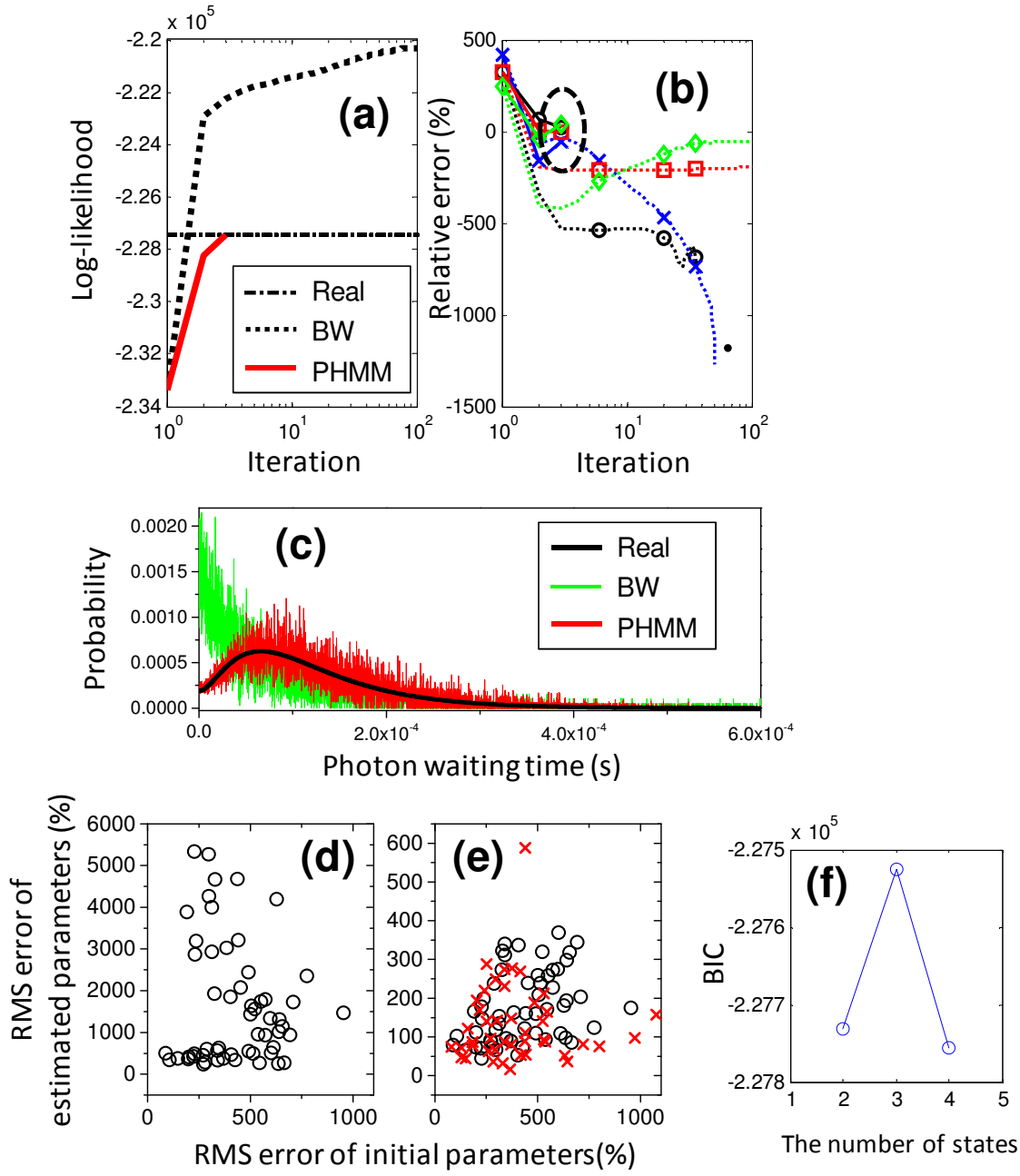

Figure 3.8 Comparison of PbPHMM and BW algorithm. (a) log-likelihood calculated by BW algorithm (black dot), PbPHMM (red solid), and real parameters (black dash-dot). (b) Relative error of photophysical parameters from BW algorithm (dotted), PbPHMM (solid). Colors are τ_{off1} (black), τ_{off2} (red), Φ_{Dark1} (blue), and Φ_{Dark2} (green). (c) Trained emission matrices by real parameters (black), BW algorithm (green), and PbPHMM (red). Root-mean-squared sum of relative errors of photophysical parameters by (d) BW and (e) PbPHMM. Red crosses denote initial parameters that led to violate photophysical constraint after BW algorithm, but resulted in relevant parameters by PbPHMM. (f) BIC calculated at 1~3 hidden states.

3.4.3 Fitting performance of PbPHMM

Fitting performance of PbPHMM including model order selection and parameter estimation accuracy was tested using simulated photons and adjusting both experimental and photophysical parameters. In order to simulate realistic photon time trace, detection efficiency was varied from 0.05 to 1 and background level was from 0 count/s to 5000 counts/s. Fluorescence quantum yield was 0.5, which is close to the value of commercially available dyes. The primary excitation intensity was controlled from 250 to 1250 W/cm^2. The number of excitation/de-excitation cycles of fluorescent probes was fixed at 10^6 except for evaluating the effect of the length of photon time traces. Approximately 30,000 photons were generated when Φ_{Det}=0.05, I_{prim}=500 W/cm^2, and noise level=1000/s.

Fitting results for photon traces generated under various detection efficiencies and noise levels are shown in Figure 3.7. From 50 time traces generated at different Φ_{Det} and noise levels, the correct number of hidden states could be determined by maximized BIC with PbPHMM as shown in Figure 3.7(a) except when Φ_{Det}=0.05 and the noise level is high (3000~5000 /s). In Figure 3.7(b), parameter estimation results of $Dark_1$ state demonstrated reduced relative errors at larger detection efficiencies and lower noise levels. It should be noted that fitting results of quantum yield and lifetime of $Dark_2$ showed the opposite tendency. Relative error of Φ_{Dark2} with no noise is larger than that of quantum yields with nonzero noise photons. The larger error is attributed to the difference in the amount of available photon waiting time information. With zero noise, ~1000 photons will show transitions into $Dark_2$. According to the definition of hidden state of photons in 3.2.1 and 3.4.1, background photons can be categorized into individual

hidden states. When the noise level increased from 1000 to 5000/s, the number of photons detected after transitions into $Dark_2$ increased from ~2000 to ~6000. However, higher noise diminishes the differences in the mean values of different states, making state assignments less accurate. Parameter estimation can be improved if longer fluorescence time traces are obtained or multiple datasets are analyzed, and will be discussed later. The reason for better parameter estimation at larger Φ_{Det} is clear; more information is available to estimate the parameters because the number of photons per excitation increased. The population of photons at state 1 and state 2 increased from 14,000 and 10,000 when Φ_{Det} =0.05 to 480,000 and 20,000 when Φ_{Det} =1.

Compared to autocorrelation methods in Figure 3.7(c), PbPHMM showed lower relative error and smaller deviation except Φ_{Dark2} with no background. Additionally PbPHMM could determine the number of dark states perfectly at low noise level (<1000/s) and with 50% accuracy when noise level=3000/s. As mentioned before, the autocorrelation method for calculating on-and off-time is only available when the underlying state connectivity and photophysical model are known.

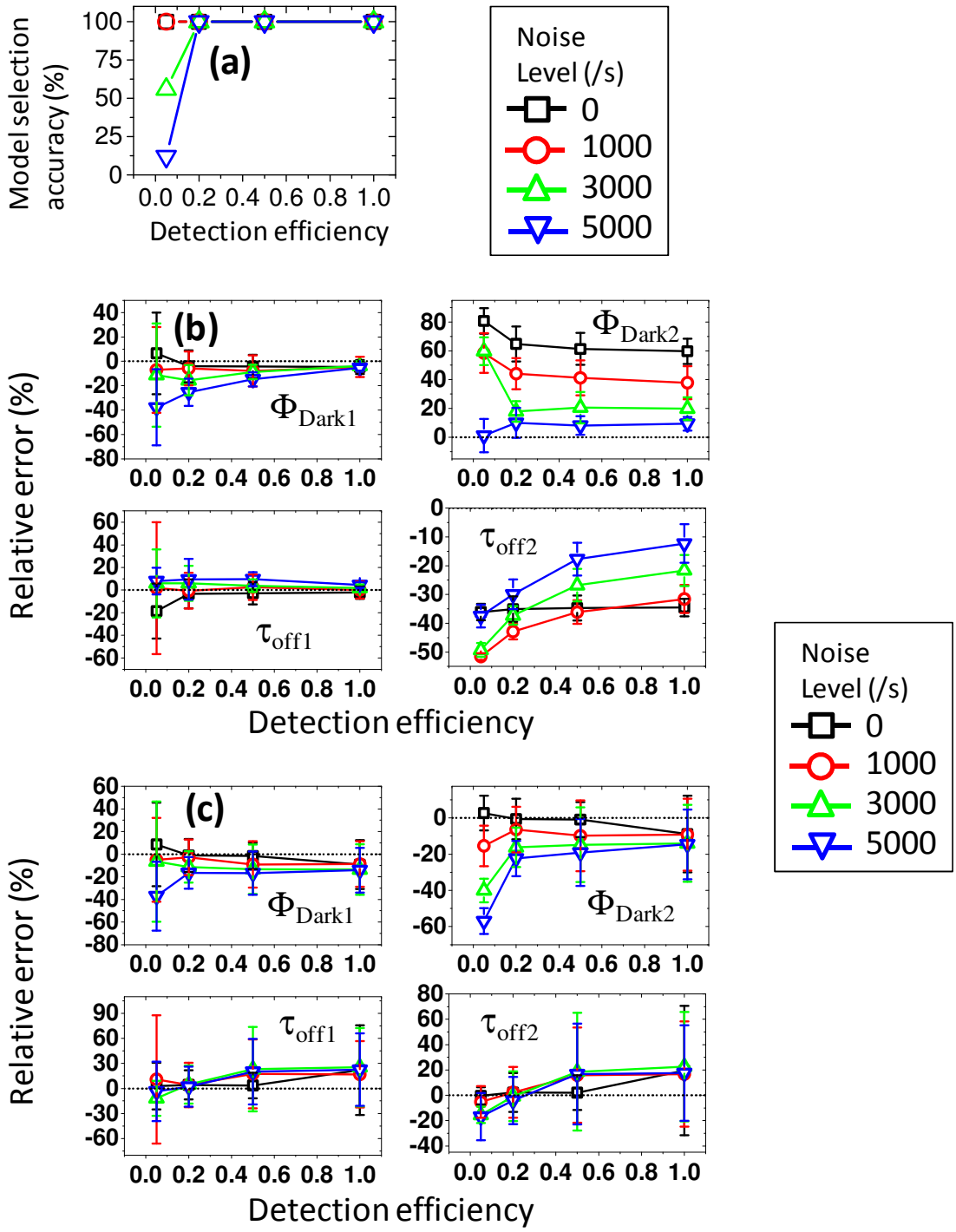

Figure 3.9 Fitting results for PbPHMM of simulated photons with Φ_{FL} =0.5 at various detection efficiencies and background levels (/s). Fifty trajectories were generated at each detection efficiency and noise level. (a) The accuracies to find the correct number of hidden state by BIC are shown in percentage. (b) Relative errors of extracted quantum yield of short (Φ_{Dark1}) and long dark state (Φ_{Dark1}) and lifetime (τ_{off1}, τ_{off2}) were calculated by eq. (3.28). (c) Relative errors calculated by autocorrelation.

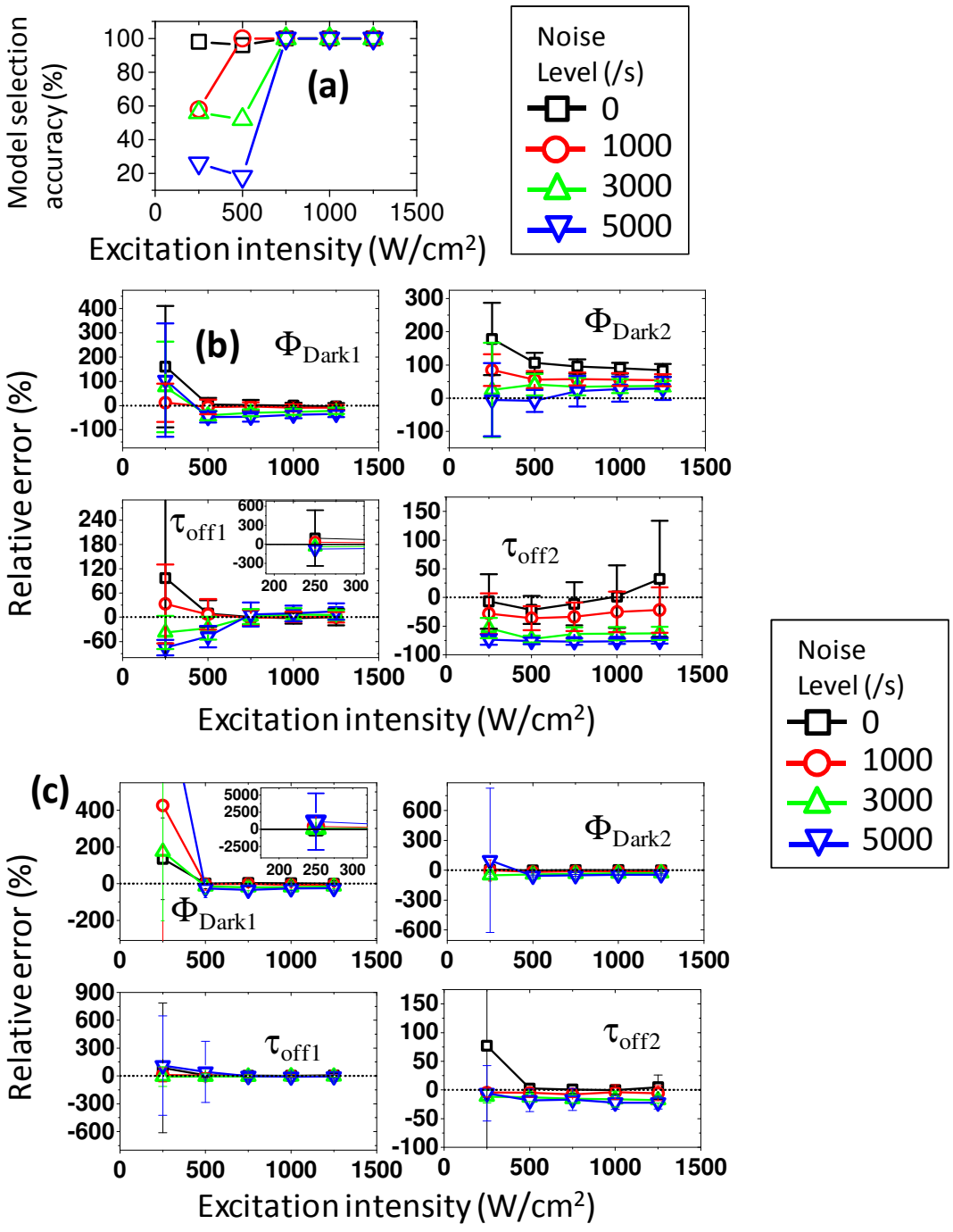

Figure 3.10 Fitting results for PbPHMM of simulated photons at various excitation intensities and background levels (/s). Detection efficiency was fixed at 0.05. Fifty trajectories were generated at each excitation intensity and noise level. (a) The accuracies to find the correct number of hidden state by BIC are shown in percentage. (b) Relative errors of extracted quantum yield of short (Φ_{Dark1}) and long dark state (Φ_{Dark1}) and lifetime (τ_{off1}, τ_{off2}) were calculated by eq. (3.28). (c) Relative errors calculated by autocorrelation.

Next, the impact of excitation intensity on fitting performance was investigated. 50 time traces were generated when the excitation intensity was varied from 250 to 1,250 W/cm^2 at different noise level. Other parameters were kept constant including Φ_{Det} =0.05, Φ_{Dark1} =0.02, Φ_{Dark2} =0.001, τ_{off1}=20 μs, and τ_{off2}=1 ms. Fitting results of traces at different excitation intensity in Figure 3.8(a) shows low model selection accuracy at low excitation intensity (250 W/cm^2).

As shown in Figure 3.8(b), photophysical parameters could be extracted more accurately at higher detection intensities. High excitation intensity shortens dwell time of fluorescent probes at ground photophysical state. Consequently the relative difference between photon waiting time at different states increases. Larger differences in photon waiting times between the states of photons lead to efficient state discrimination, improving fitting performance of PbPHMM. For example, given photophysical parameters and Φ_{Det}=0.05 with no background, the ratio of mean photon waiting time at state 2 to state 1, $\Delta t_2/\Delta t_1$, and the ratio of state 3 to state 2, $\Delta t_3/\Delta t_2$, were 3.3 and 5.7 respectively. Those ratios increased to $\Delta t_2/\Delta t_1$=5.2 and $\Delta t_3/\Delta t_2$=15.0 at the excitation intensity of 1000 W cm^{-2}, which improved parameter estimation and model selection accuracy.

Fitting performance at various dark state quantum yields was studied. As presented in Figure 3.9(a), model order selection accuracy was lower than 50%, showing an adverse impact of background noise and insufficient population of dark state. The impact was mitigated by an increased amount of information about Dark$_1$ state when Φ_{Dark1} >0.05. Accuracy larger than 90% was achieved for Φ_{Dark1} > 0.05. Φ_{Dark1} and τ_{off1} could be more accurately estimated at larger Φ_{Dark1} and lower noise level. Quantum yield and lifetime of

$Dark_2$ did not show systematic dependency on Φ_{Dark1}. Relative error showed the same tendency as parameter estimation in varying Φ_{Det} and I_{prim} in Figure 3.7(b) and Figure 3.8(b), respectively; Background noise had a positive impact on estimating quantum yield and deteriorated lifetime estimation. Average relative errors of the parameters from PbPHMM and autocorrelation method are similar to each other, but PbPHMM showed less deviation.

Dark state lifetime showed a distinct impact on fitting performance (Figure 3.10). The lifetime of $Dark_1$ state was varied from 20 μs to 2 ms while the lifetime of $Dark_2$ state was fixed at 2 ms. As shown in Figure 3.10(a), model order selection accuracy decreased as lifetimes of two dark states became similar. When $\tau_{off1}>5\times10^{-4}$ s with nonzero background, most of fitting results showed that only one dark state existed. In this condition, as presented in Figure 3.10(b), photon waiting times at state 2, Δt_2, and state 3, Δt_3, became similar, showing the ratio $\Delta t_3/\Delta t_2$ was reduced to less than 2. PbPHMM could not detect two dark states, and BIC was maximized at two hidden states in most of data sets with $\Delta t_3/\Delta t_2<2$. Although the model selection accuracy when τ_{off1}=0.5 and 1 ms with noise level = 5000 counts/s noise was 40% and 60% respectively, relative errors of photophysical parameters became extremely large as shown in Figure 3.10(c). Autocorrelation analysis in Figure 3.10(d) also showed horrible parameter estimation when $\Delta t_3/\Delta t_2<2$.

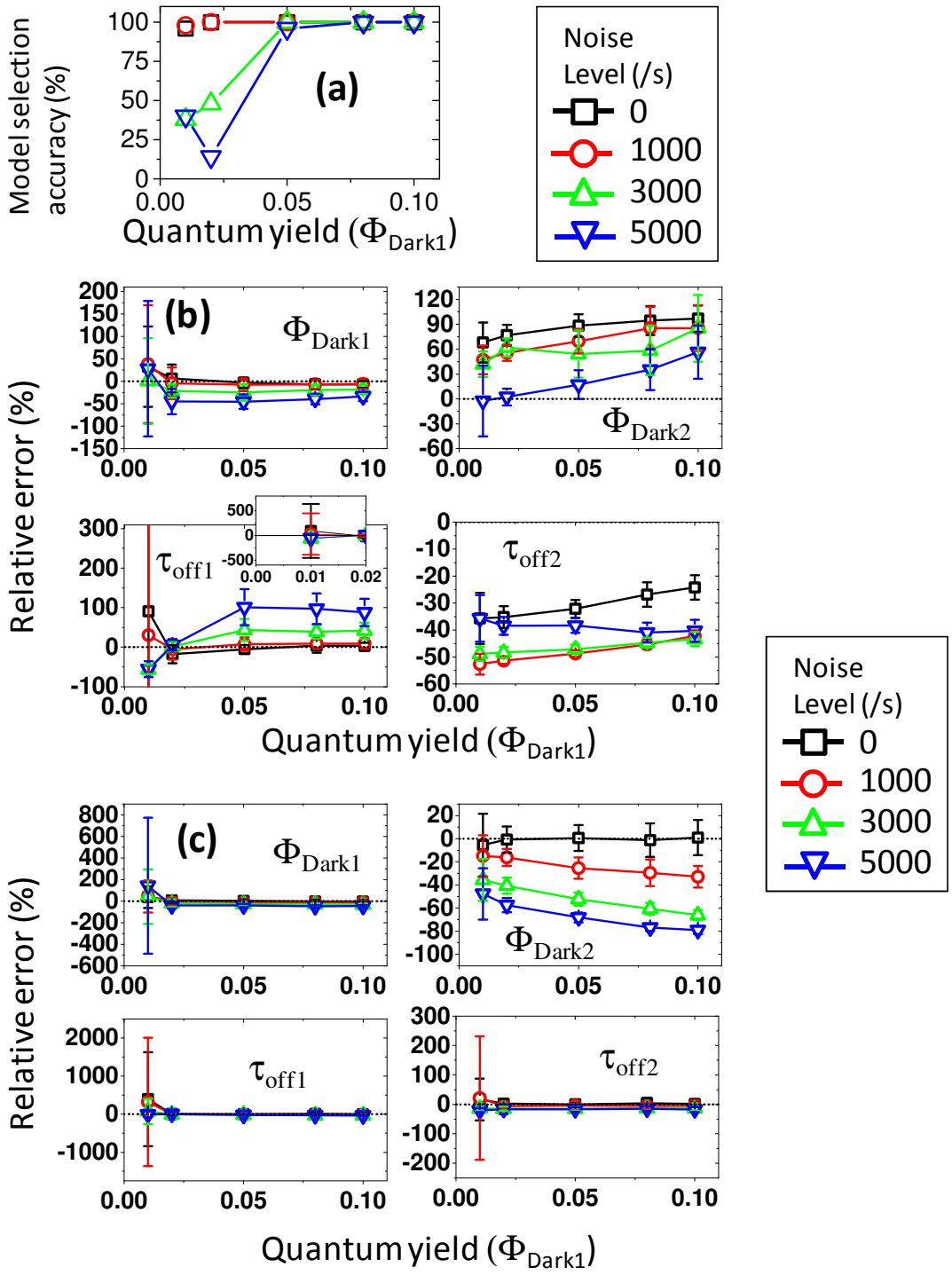

Figure 3.11 Fitting results for PbPHMM of simulated photons at various quantum yields of the shorter dark state and background levels (/s). Detection efficiency was fixed at 0.05. Fifty trajectories were generated at each dark state quantum yield and noise level. (a) The accuracies to find the correct number of hidden state by BIC are shown in percentage. (b) Relative errors of extracted quantum yield of short (Φ_{Dark1}) and long dark state (Φ_{Dark1}) and lifetime (τ_{off1}, τ_{off2}) were calculated by eq. (3.28). (c) Relative errors calculated by autocorrelation.

Fitting results from longer time traces clearly showed impact of noise level on fitting performance and removes the opposite tendency in estimating the parameters of $Dark_2$. Figure 3.11 shows the fitting performance with the various numbers of photons (N_p). N_p was adjusted by changing the number of excitation/de-excitation cycles from 1×10^6 to 4×10^7 with the fixed Φ_{Det} =0.05 and different noise level. Fluorescent probes with high photostability can endure larger number of excitation/de-excitation cycles. It is reported that photostability of fluorescent probes can be enhanced by 10 fold by applying additives combined with an oxygen-scavenging system [52, 60] or increased 20~75 fold by linking protective agents [61-63]. Novel fluorophores including DNA-encapsulated silver nanodot and semiconductor quantum dots demonstrated 10^3~10^4 fold increased photostability [64, 65]. Figure 3.11 shows that the photophysical model of these photostable probes can be estimated more accurately than from short fluorescent time traces of conventional dyes. In Figure 3.11(a), PbPHMM showed robust model order selection accuracy along a range of N_p at low noise level (0 and 1000/s). Model selection accuracy with noise level=3000/s was larger than 60% when N_p <200,000. The accuracy, however, decreased below 60% when fluorescent time traces of more than 200,000 photons were analyzed. Moreover, when noise level increased to 5000/s, the PbPHMM method could not predict the correct number of hidden state. The reason of low accuracy when noise level=5000/s is the same as previous analysis, that is, the background noise induces indiscriminable photon waiting times. PbPHMM predicted two hidden states when more than 10^5 photons were analyzed with noise level=5000/s.

The low accuracy when noise level=3000/s can be explained in a different way. When N_p >200,000, PbPHMM predicted 40~50% of data traces generated from 4 hidden

states. A hidden photophysical state, which did not exist in photon-generating model, was predicted to have a lifetime similar to the inverse of noise level (6×10^{-4} s) and quantum yield close to one of $Dark_2$ (7×10^{-4}). Additional hidden states increase the likelihood of observed data, as mentioned in Section 1.5.2.4. The increased likelihood, caused by adding this state, exceeded that of penalizing term in eq. 2.1. Therefore BIC was maximized at 4 hidden states.

Photophysical parameters could be estimated with relative error less than 50% when more than 100,000 photons are analyzed. Upper right panels in Figure 3.7(b), 3.8(b), and 3.9(b) showed a contradictory, larger relative error of Φ_{Dark2} at lower noise level. Those relative errors became relevant as N_p increased to 10^6; parameter estimation improved at lower noise levels, and the relative error reduced to less than 20% when longer time traces were analyzed due to sufficient transitions to $Dark_2$. Φ_{Dark2} were underestimated even from the analysis of long time traces. The underestimation was due to a false-negative in a trained transition probability into state 2, which was caused by reduced differences in mean photon waiting time in state 1 and state 2.

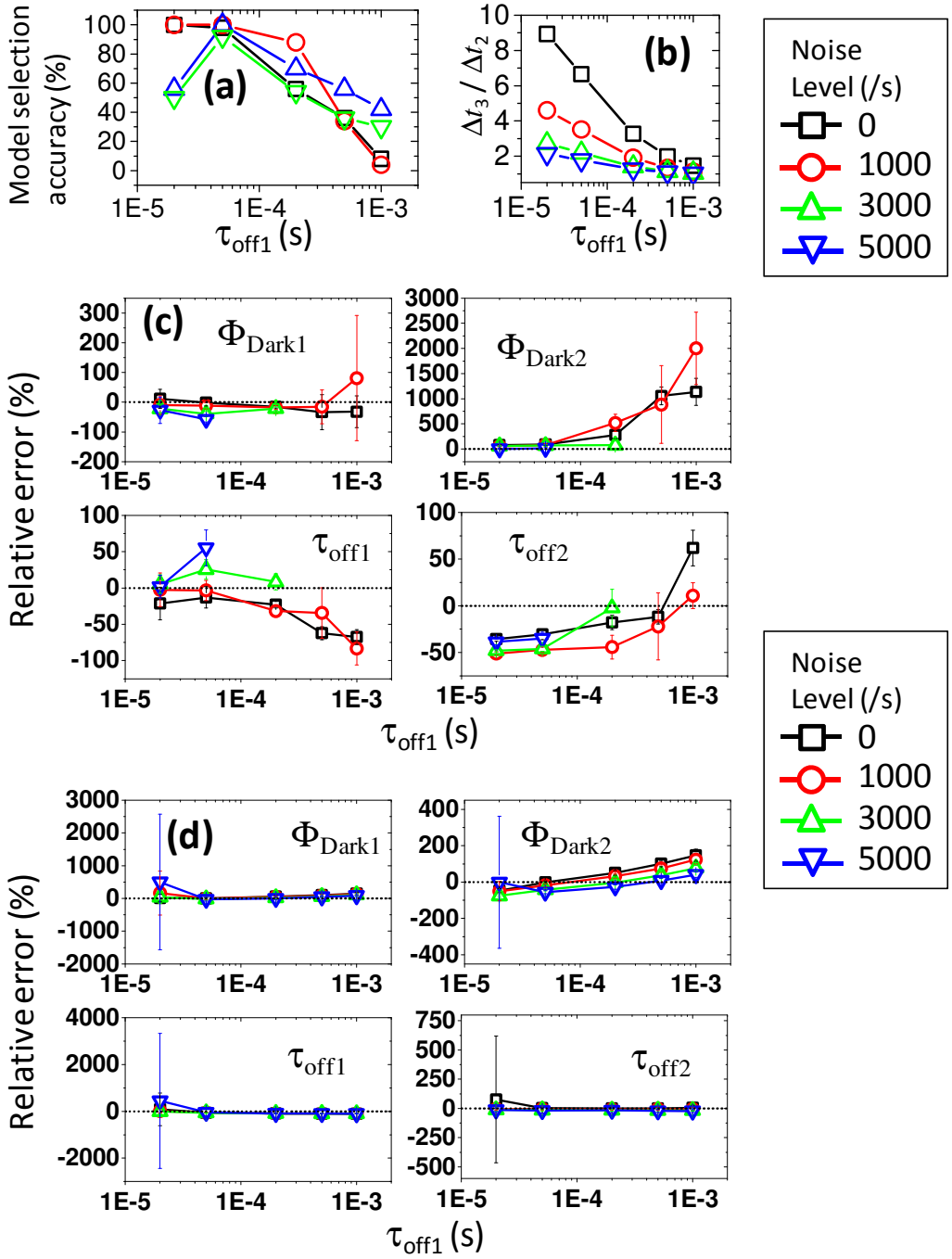

Figure 3.12 Fitting results for PbPHMM of simulated photons at various lifetimes of the shorter dark state and background levels (/s). Detection efficiency was fixed at 0.05. Fifty trajectories were generated at each lifetime and noise level. (a) The accuracies to find the correct number of hidden state by BIC are shown in percentage. (b) The ratio of Δt_3 to Δt_2 at different background levels. (c) Relative errors of extracted quantum yield of short (Φ_{Dark1}) and long dark state (Φ_{Dark1}) and lifetime (τ_{off1}, τ_{off2}) were calculated by eq. (3.28). (d) Relative errors calculated by autocorrelation.

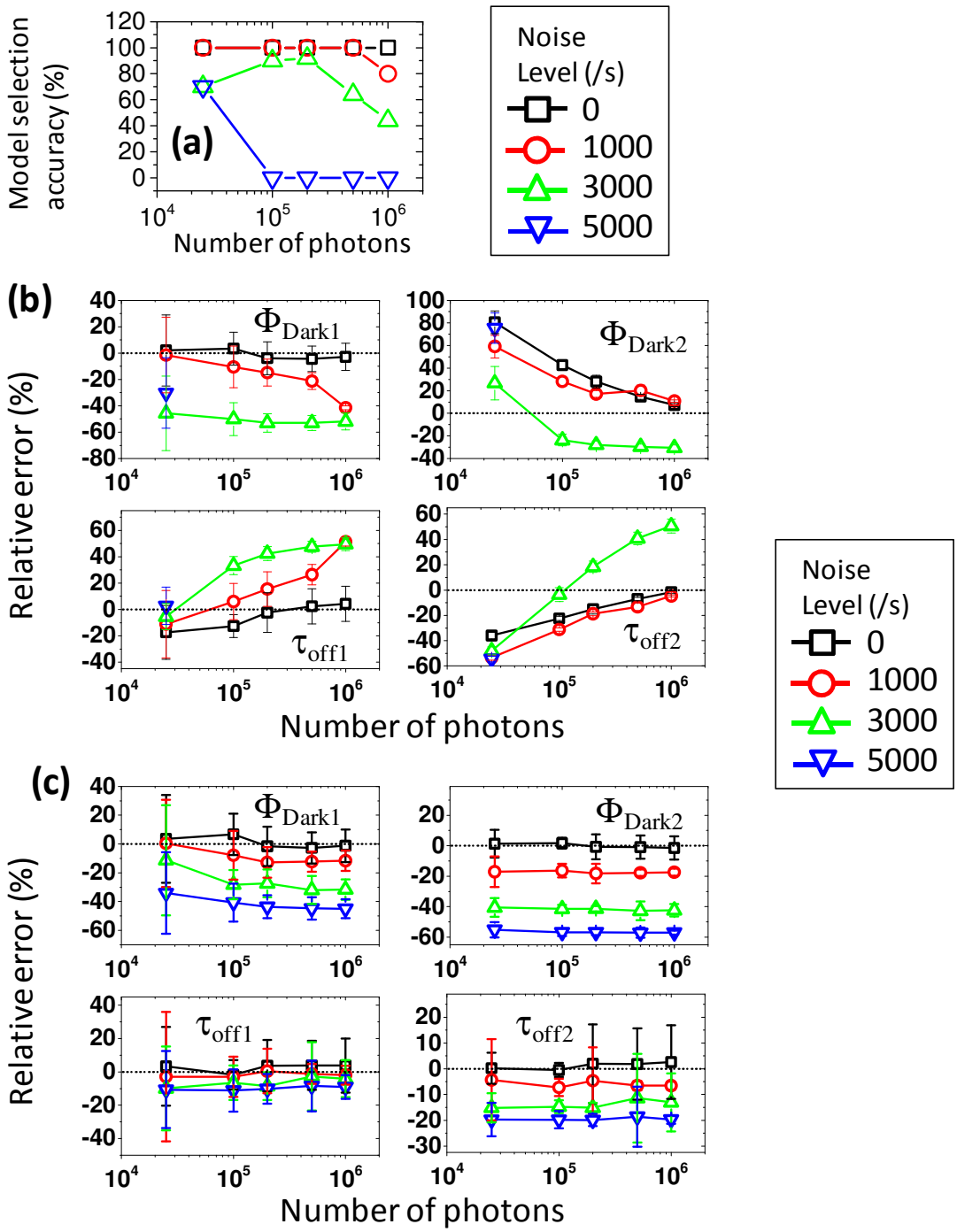

Figure 3.13 Fitting results for PbPHMM of simulated photons at different number of photons and background levels (/s). Detection efficiency was fixed at 0.05. Fifty trajectories were generated with each number of photons and noise level. The accuracies to find the correct number of hidden state by BIC (a) are shown in percentage. (b) Relative errors of extracted quantum yield of short (Φ_{Dark1}) and long dark state (Φ_{Dark1}) and lifetime (τ_{off1}, τ_{off2}) were calculated from different length of photon time traces. (c) Relative errors calculated by autocorrelation.

3.4.4 Fitting multiple observation sequences

Even though PbPHMM showed good model selection accuracy and acceptable parameter estimation from short time traces, the relative errors from short traces increased when the noise level is not negligible (>1000/s). Specifically, parameter estimation of less populated states is severely influenced by background noise. Analysis of longer time traces alleviated poor fitting quality. However, collecting sufficient number of photons is allowed only if the fluorescent probe has a sufficient photostability, and the relative error in parameter estimation is not negligible.

The alternative way to accurately estimate model parameters with HMM is re-estimating the parameters from independent multiple observations [66]. Re-estimation has been used to obtain more accurate parameters in speech recognition [47], handwriting recognition [67], and FRET analysis [10, 13, 14]. Applying these studies, the new transition and emission matrices were estimated, averaging the matrices over multiple short time traces using equations 1.8, 1.12, and 3.26.

Figure 3.12 and 3.13 shows relative errors of photophysical parameters calculated from the re-estimated transition and emission matrices over the same time traces as in 3.4.3. Figure 3.12 (a) and (b) show the impact of Φ_{Det} and I_{prim} respectively. Relative errors as a function of Φ_{Dark1} and τ_{off1} are shown in Figure 3.13(a) and (b). All four panels showed parameter estimation better than in fitting individual short or long time traces. Even the difference between and τ_{off1} and τ_{off2} is small (τ_{off1} =5×10^{-4} and 1×10^{-3}), the relative errors were improved than fitting individual traces. Unit weight factor was used in averaging forward and backward probabilities over multiple traces in eq. 3.26, as reported to achieve the best performances [13, 47]. In conclusion, parameter estimation

can be improved if multiple independent trajectories are available, which is especially useful when the length of time trace is limited.

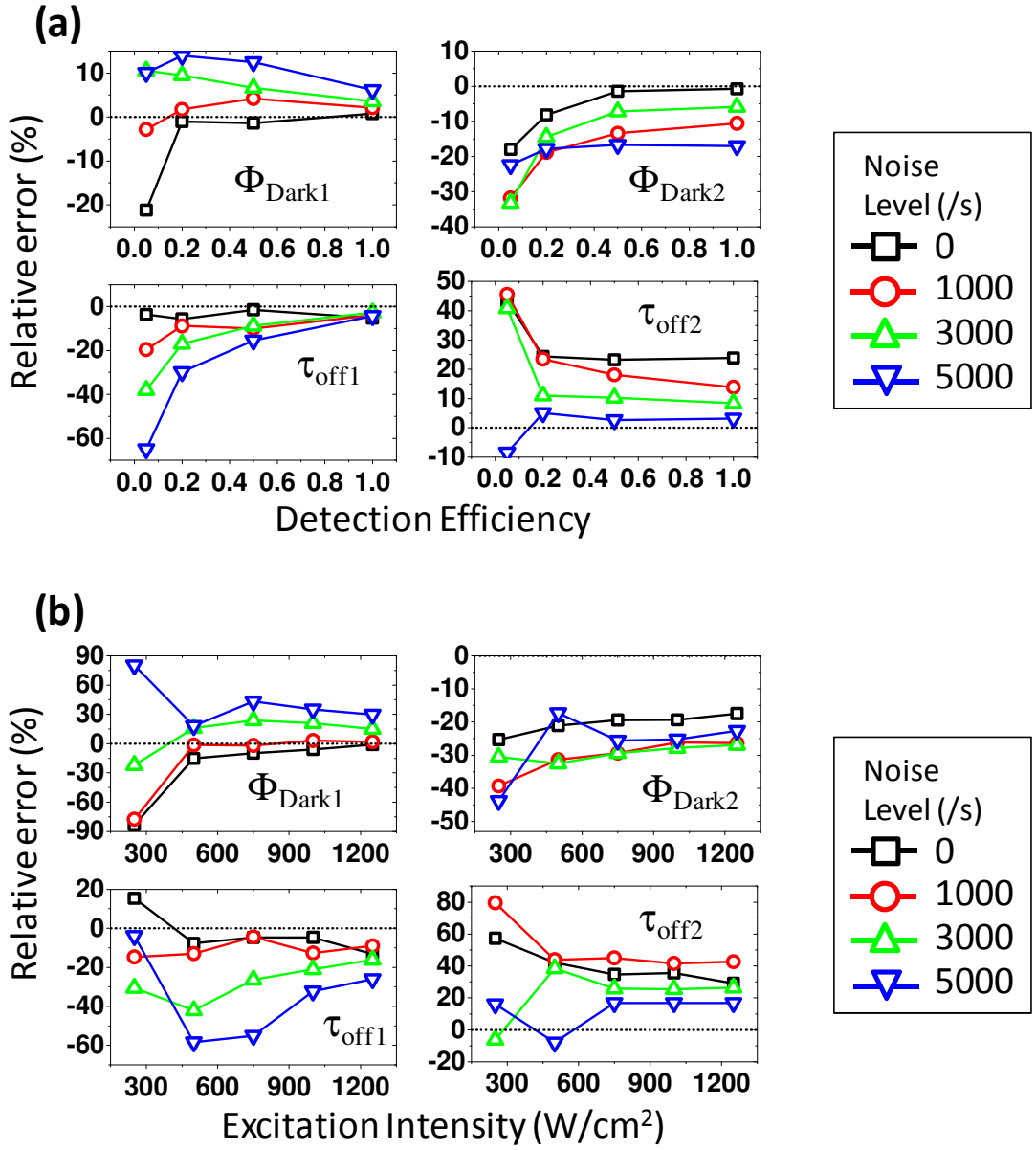

Figure 3.14 Relative errors of re-estimated parameters from multiple observation sequences. Effect of (a) detection efficiency and (b) excitation intensity.

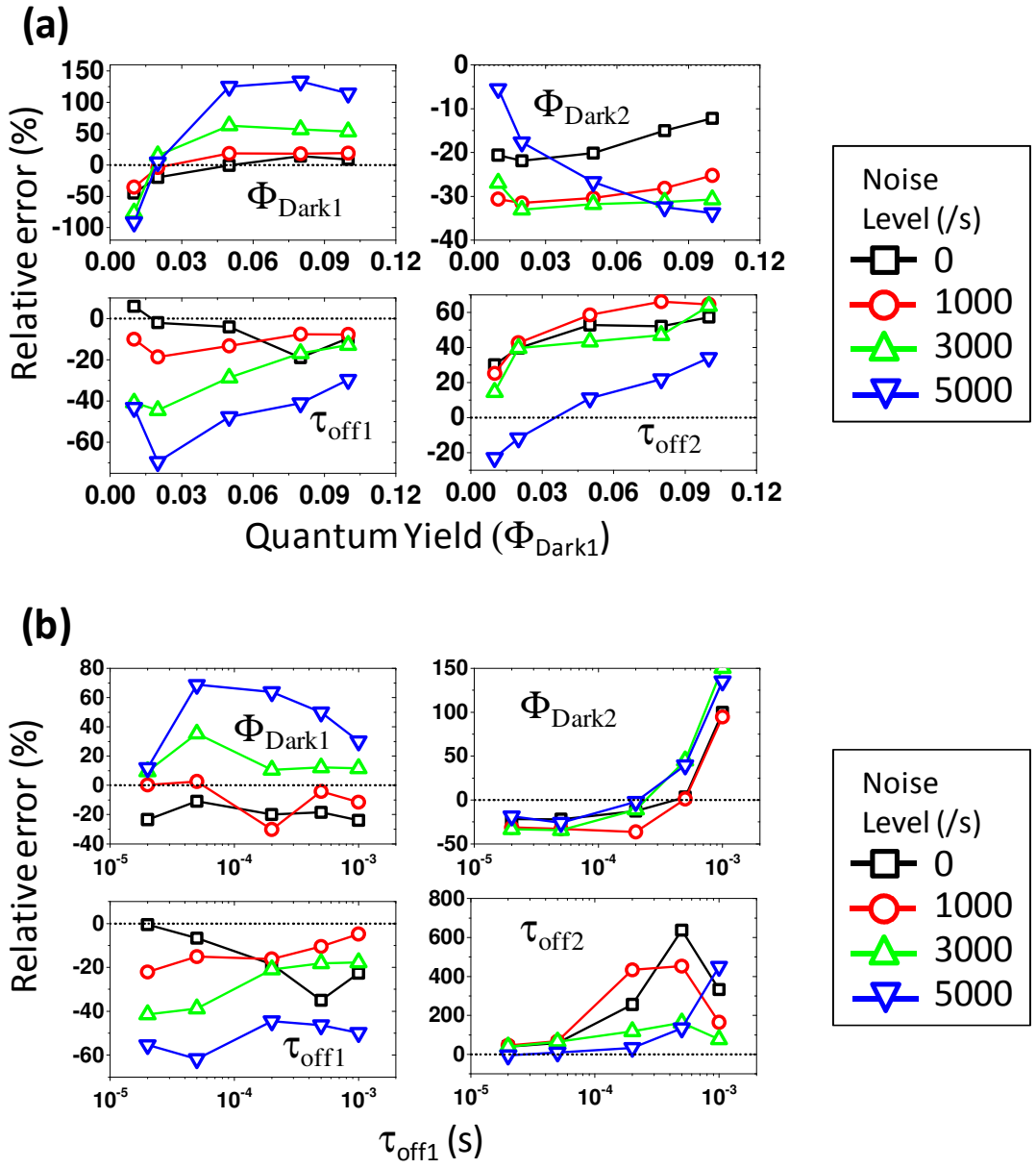

Figure 3.15 Relative errors of re-estimated parameters from multiple observation sequences. Effect of (a) dark state quantum yield and (b) lifetime.

3.4.5 Comparison of HMM to other methods

In this section, I compare fitting performance of PbPHMM to other stochastic methods, such as change point detection (CPD) [68] and Markov-modulated Poisson processes (MMPP) [69].

CPD method determines the temporal positions of intensity change points by comparing hypothesis of change point existence. This method was applied to find transitions between states defined in section 3.3.3. Simulated photons with Φ_{Det}=0.05 and 1 and background level=0~15000 (counts/s) were analyzed to find mean intensity and the number of hidden states. One of the disadvantages of CPD method over PbPHMM was low model order selection accuracy. When detection efficiency was 0.05 with no background, CPD method determined correct number of hidden states. With background noise, however, CPD determined fewer hidden states than were present with low detection efficiency (Φ_{Det} =0.05) and more hidden states at unity detection efficiency (Figure 3.14(a)). Additionally, CPD produces higher relative error in photon waiting times than PbPHMM. At Φ_{Det}=0.05 and no background, relative errors of photon waiting time by CPD were 1.3, -45, and -87 % for state 1, 2, and 3, respectively. PbPHMM resulted in relative errors for photon waiting times -12, -3.5, and -2.8 %, showing less overall relative error. Finally, CPD is more computationally demanding, requiring more than 24 hours to analyze 300k photons. Figure 3.14(b) shows root-mean-squared sum of relative errors at various detection efficiency and background level.

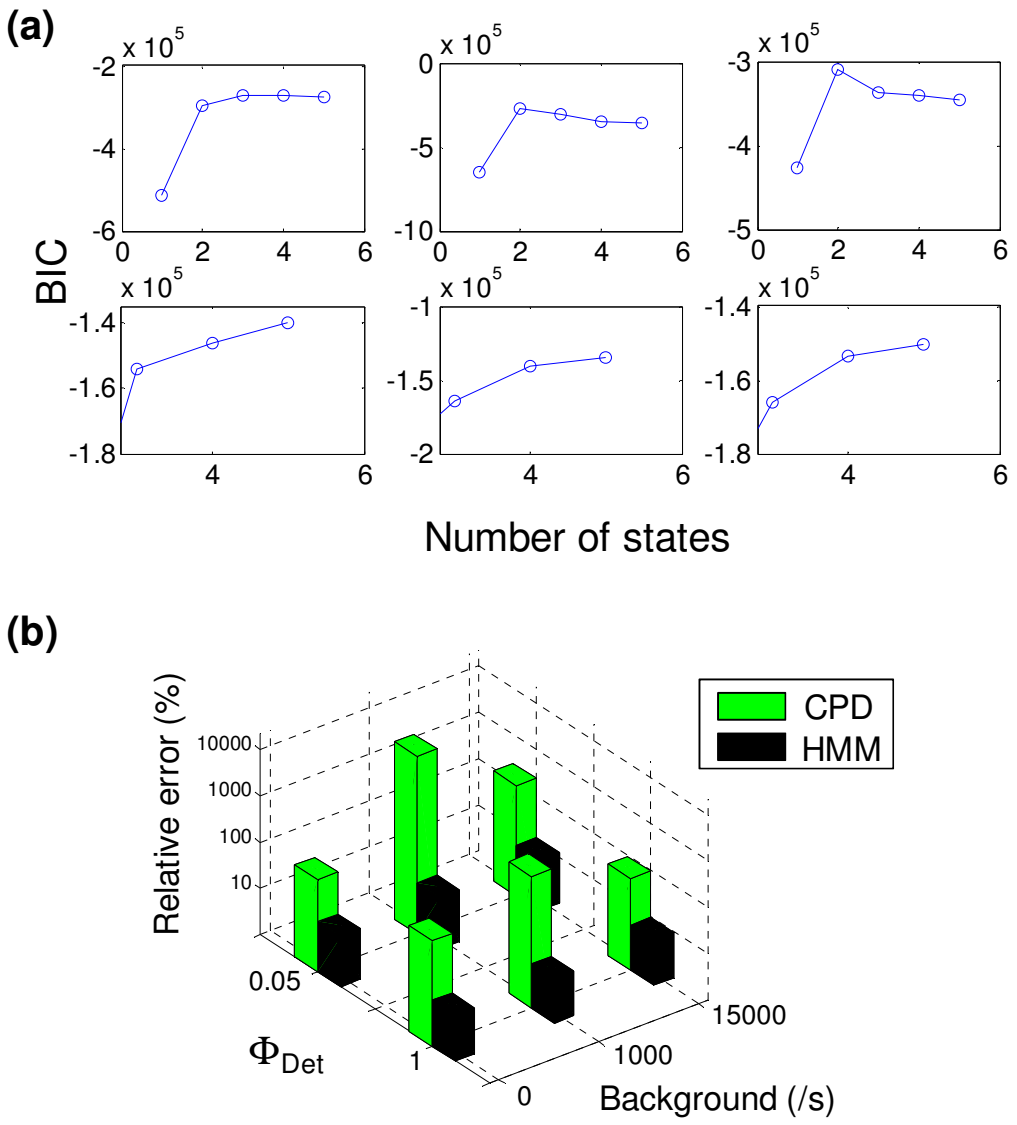

Figure 3.16 Fitting performance by CPD and PbPHMM. (a) BIC was calculated by CPD at 1~5 hidden states. The analyses were done for 300k photons collected with detection efficiency 0.05 (first row) and 1 (second row) and background level 0 (first column), 1000 (second column), and 5000 counts/s (third column). Figures at second row were zoomed around 3~5 states to clarify the difference of BIC values. (b) Root mean square of relative errors of photon waiting times at each state extracted from CPD and PbPHMM at different detection efficiency and background level.

The fitting algorithm for MMPP was implemented in R package HIDDENMARKOV [70]. Short trajectories with ~10k exponential random numbers could be analyzed with the package. However, fitting was terminated due to numerical error while analyzing longer realistic data (>100k photons) with low detection efficiency.

3.5 Conclusion

PBPHMM was developed to build a photophysical model from non-Poisson distributed photon time traces, with the goal of incorporating the information often discarded by time-binning the fluorescence signal. PbPHMM has unlimited time resolution in estimating parameters, since it analyzes photon waiting times, not time-binned intensity. Additionally, in PbPHMM, the solution of the master equation is not required to extract the rates of underlying dynamics, which is often long and complicated. The relation between photophysical parameters and model parameters was formulated. The formula was also used in constraining transition and emission probabilities to be photophysically relevant. By using PbPHMM, fitting iterations could achieve maximum likelihood complying photophysical constraints and extract photophysical parameters with low relative error. Fitting performance of PbPHMM was investigated by analyzing photon time traces generated from various conditions. Model order selection accuracy was kept >90%, and photophysical parameters could be estimated with relative error less than 50% for realistic conditions (Φ_{Det} = 0.05, noise level <3000/s, and I_{prim}>500 W/cm^2). When multiple time traces are available, the fitting performance of PbPHMM could be improved further, showing relative error less than 10%. Fitting performance evaluation at various conditions demonstrated that PbPHMM will be an adequate tool for single

molecule data analysis. PBPHMM showed better fitting performance than other published methods including CPD and MMPP.

3.6 References

1. Hofkens, J., M. Maus, T. Gensch, T. Vosch, M. Cotlet, F. Köhn, A. Herrmann, K. Müllen, and F. De Schryver, *Probing photophysical processes in individual multic hromophoric dendrimers by single-molecule spectroscopy.* Journal of the America n Chemical Society, 2000. **122**(38): p. 9278-9288.

2. Hernando, J., d.S.M. van, D.E.M.H.P. van, M. Sauer, M.F. Garcia-Parajo, and H.N .F. van, *Excitonic Behavior of Rhodamine Dimers: A Single-Molecule Study.* The J ournal of Physical Chemistry A, 2003. **107**(Copyright (C) 2012 American Chemic al Society (ACS). All Rights Reserved.): p. 43-52.

3. Bockenhauer, S., A. Fürstenberg, X.J. Yao, B.K. Kobilka, and W.E. Moerner, *Conf ormational Dynamics of Single G Protein-Coupled Receptors in Solution.* The Jou rnal of Physical Chemistry B, 2011. **115**(45): p. 13328-13338.

4. Kask, P., K. Palo, D. Ullmann, and K. Gall, *Fluorescence-Intensity Distribution A nalysis and Its Application in Biomolecular Detection Technology.* Proceedings of the National Academy of Sciences of the United States of America, 1999. **96**(24): p. 13756-13761.

5. Wang, J. and P. Wolynes, *Intermittency of Single Molecule Reaction Dynamics in Fluctuating Environments.* Physical Review Letters, 1995. **74**(21): p. 4317.

6. Rabiner, L.R., B.H. Juang, S.E. Levinson, and M.M. Sondhi, *Recognition of Isolat ed Digits Using Hidden Markov Models With Continuous Mixture Densities.* AT& T Technical Journal, 1985. **64**(6): p. 1211-1234.

7. Chung, S.H., J.B. Moore, L. Xia, L.S. Premkumar, and P.W. Gage, *Characterizati on of Single Channel Currents Using Digital Signal Processing Techniques Based on Hidden Markov Models.* Philosophical Transactions: Biological Sciences, 199 0. **329**(1254): p. 265-285.

8. Qin, F., A. Auerbach, and F. Sachs, *A Direct Optimization Approach to Hidden M*

arkov Modeling for Single Channel Kinetics. Biophysical Journal, 2000. **79**(4): p. 1915-1927.

9. Andrec, M., R.M. Levy, and D.S. Talaga, *Direct Determination of Kinetic Rates fr om Single-Molecule Photon Arrival Trajectories Using Hidden Markov Models.* T he Journal of Physical Chemistry A, 2003. **107**(38): p. 7454-7464.

10. McKinney, S.A., C. Joo, and T. Ha, *Analysis of Single-Molecule FRET Trajectorie s Using Hidden Markov Modeling.* Biophysical Journal, 2006. **91**(5): p. 1941-195 1.

11. Jäger, M., A. Kiel, D.-P. Herten, and F.A. Hamprecht, *Analysis of Single-Molecule Fluorescence Spectroscopic Data with a Markov-Modulated Poisson Process.* Ch emPhysChem, 2009. **10**(14): p. 2486-2495.

12. Bilmes, J.A., *What HMMs can do.* IEICE TRANSACTIONS on Information and Systems, 2006. **89**(3): p. 869-891.

13. Lee, T.-H., *Extracting Kinetics Information from Single-Molecule Fluorescence R esonance Energy Transfer Data Using Hidden Markov Models.* The Journal of Ph ysical Chemistry B, 2009. **113**(33): p. 11535-11542.

14. Liu, Y., J. Park, K.A. Dahmen, Y.R. Chemla, and T. Ha, *A Comparative Study of Multivariate and Univariate Hidden Markov Modelings in Time-Binned Single-M olecule FRET Data Analysis.* The Journal of Physical Chemistry B, 2010. **114**(16) : p. 5386-5403.

15. Uphoff, S., K. Gryte, G. Evans, and A.N. Kapanidis, *Improved Temporal Resoluti on and Linked Hidden Markov Modeling for Switchable Single-Molecule FRET.* C hemPhysChem, 2011. **12**(3): p. 571-579.

16. Messina, T.C., H. Kim, J.T. Giurleo, and D.S. Talaga, *Hidden Markov Model Anal ysis of Multichromophore Photobleaching.* The Journal of Physical Chemistry B, 2006. **110**(33): p. 16366-16376.

17. Chung, S.H., V. Krishnamurthy, and J.B. Moore, *Adaptive Processing Techniques Based on Hidden Markov Models for Characterizing Very Small Channel Current s Buried in Noise and Deterministic Interferences.* Philosophical Transactions: Bi ological Sciences, 1991. **334**(1271): p. 357-384.

18. Qin, F., A. Auerbach, and F. Sachs, *Estimating single-channel kinetic parameters f rom idealized patch-clamp data containing missed events.* Biophysical Journal, 19 96. **70**(1): p. 264-280.

19. Venkataramanan, L., R. Kuc, and F.J. Sigworth, *Identification of hidden Markov models for ion channel currents. II. State-dependent excess noise.* Signal Processi ng, IEEE Transactions on, 1998. **46**(7): p. 1916-1929.

20. Qin, F., *Restoration of Single-Channel Currents Using the Segmental k-Means Me thod Based on Hidden Markov Modeling.* Biophysical Journal, 2004. **86**(3): p. 14 88-1501.

21. Linaro, D., M. Storace, and M. Giugliano, *Accurate and Fast Simulation of Chan nel Noise in Conductance-Based Model Neurons by Diffusion Approximation.* PL oS Comput Biol, 2011. **7**(3): p. e1001102.

22. Neuert, G., C. Albrecht, E. Pamir, and H.E. Gaub, *Dynamic force spectroscopy of the digoxigenin-antibody complex.* FEBS letters, 2006. **580**(2): p. 505-509.

23. Chen, C., M.L. Juan, Y. Li, G. Maes, G. Borghs, P. Van Dorpe, and R. Quidant, *En hanced Optical Trapping and Arrangement of Nano-Objects in a Plasmonic Nano cavity.* Nano Letters, 2011. **12**(1): p. 125-132.

24. Chodera, J.D., P. Elms, F. Noe, B. Keller, C.M. Kaiser, A. Ewall-Wice, S. Marqus ee, C. Bustamante, and N.S. Hinrichs, *Bayesian hidden Markov model analysis of single-molecule force spectroscopy: characterizing kinetics under measurement u ncertainty.* arXiv.org, e-Print Arch., Condensed Matter, 2011(Copyright (C) 2012 American Chemical Society (ACS). All Rights Reserved.): p. 1-12, arXiv:1108.14 30v1101 [cond-mat.stat-mech].

25. Gao, Y., G. Sirinakis, and Y. Zhang, *Highly Anisotropic Stability and Folding Kin etics of a Single Coiled Coil Protein under Mechanical Tension.* Journal of the A merican Chemical Society, 2011. **133**(32): p. 12749-12757.

26. Gao, Y., S. Zorman, G. Gundersen, Z. Xi, L. Ma, G. Sirinakis, J.E. Rothman, and Y. Zhang, *Single Reconstituted Neuronal SNARE Complexes Zipper in Three Disti nct Stages.* Science, 2012. **337**(6100): p. 1340-1343.

27. Fichthorn, K.A. and W.H. Weinberg, *Theoretical foundations of dynamical Monte Carlo simulations.* Journal of Chemical Physics, 1991. **95**(2): p. 1090.

28. Schenter, G.K., H.P. Lu, and X.S. Xie, *Statistical Analyses and Theoretical Model s of Single-Molecule Enzymatic Dynamics.* The Journal of Physical Chemistry A, 1999. **103**(49): p. 10477-10488.

29. Baum, L.E., *An inequality and associated maximization technique in statistical est imation for probabilistic functions of Markov processes.* Inequalities, 1972. **3**: p. 1 -8.

30. Jianying, H., M.K. Brown, and W. Turin, *HMM based online handwriting recogni tion.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1996. **18**(10): p. 1039-1045.

31. Bahl, L.R., F. Jelinek, and R. Mercer, *A Maximum Likelihood Approach to Contin uous Speech Recognition.* Pattern Analysis and Machine Intelligence, IEEE Trans actions on, 1983. **PAMI-5**(2): p. 179-190.

32. Jung, S. and R.M. Dickson, *Hidden Markov Analysis of Short Single Molecule Int ensity Trajectories.* J. Phys. Chem. B, 2009. **113**(42): p. 13886-13890.

33. Noé, F., *Probability distributions of molecular observables computed from Marko v models.* The Journal of Chemical Physics, 2008. **128**(24): p. 244103.

34. Papoulis, A. and S.U. Pillai, *Probability, random variables, and stochastic proces ses*. 4th ed. 2002, Boston: McGraw-Hill. x, 852 p.

35. Devore, J.L., *Probability and statistics for engineering and the sciences*. 6th ed. 2 004, Belmont, CA: Thomson-Brooks/Cole. xvi, 795 p.

36. Horowitz, P. and W. Hill, *The art of electronics*. 2nd ed. 1989, Cambridge Englan d ; New York: Cambridge University Press. xxiii, 1125 p.

37. *Becker & Hickl GmbH, Routing modules for time-correlated single photon counti ng, manual*. 2000.

38. Cooper, R.B., *Introduction to queueing theory*. 2d ed. 1981, New York: North Hol land. xv, 347 p.

39. Kapur, K.C. and M. Pecht, *Reliability engineering*. 2014, Hoboken, New Jersey:

Wiley. xix, 489 pages.

40. Cox, D.R. and W.L. Smith, *On the Superposition of Renewal Processes.* Biometri ka, 1954. **41**(1/2): p. 91-99.

41. Hoopen, M.T. and H.A. Reuver, *The Superposition of Random Sequences of Event s.* Biometrika, 1966. **53**(3/4): p. 383-389.

42. Whitt, W., *Approximating a Point Process by a Renewal Process, I: Two Basic Me thods.* Operations Research, 1982. **30**(1): p. 125-147.

43. Kallen, M., R. Nicolai, and S. Farahani, *Superposition of renewal processes for m odelling imperfect maintenance.* Risk, Safety and Reliability, 2010: p. 629-634.

44. Birolini, A., *Reliability engineering : theory and practice*. 7th ed. 2013, Heidelber g ; New York: Springer. xv, 626 p.

45. Rabiner, L.R., *A tutorial on hidden Markov models and selected applications in sp eech recognition.* Proceedings of the IEEE, 1989. **77**(2): p. 257-286.

46. Xiaolin, L., M. Parizeau, and R. Plamondon, *Training hidden Markov models with multiple observations-a combinatorial method.* IEEE Transactions on Pattern Ana lysis and Machine Intelligence, 2000. **22**(4): p. 371-377.

47. Davis, R.I., B.C. Lovell, and T. Caelli. *Improved estimation of hidden Markov mo del parameters from multiple observation sequences*. in *Pattern Recognition, 200 2. Proceedings. 16th International Conference on*. 2002.

48. Gillespie, D.T., *A general method for numerically simulating the stochastic time e volution of coupled chemical reactions.* Journal of Computational Physics, 1976. **22**(4): p. 403-434.

49. Gillespie, D.T., *Exact stochastic simulation of coupled chemical reactions.* The Jo urnal of Physical Chemistry, 1977. **81**(25): p. 2340-2361.

50. Dickson, R.M., A.B. Cubitt, R.Y. Tsien, and W.E. Moerner, *On/off blinking and s witching behavior of single molecules of green fluorescent protein.* Nature (Londo n), 1997. **388**(6640): p. 355-358.

51. Levitus, M. and S. Ranjit, *Cyanine dyes in biophysical research: the photophysics of polymethine fluorescent dyes in biomolecular environments.* Quarterly Reviews of Biophysics, 2011. **44**(01): p. 123-151.

52. Eggeling, C., J. Widengren, R. Rigler, and C.A.M. Seidel, *Photostability of Fluorescent Dyes for Single-Molecule Spectroscopy: Mechanisms and Experimental Methods for Estimating Photobleaching in Aqueous Solution*, in *Applied Fluorescence in Chemistry, Biology and Medicine*. 1999, Springer Berlin Heidelberg. p. 193-240.

53. Walder, C.J., P.J. Kootsookos, and B.C. Lovell. *Towards a maximum entropy method for estimating HMM parameters*. in *Workshop on Digital Image Computing*. 2003. Australian Pattern Recognition Society.

54. Durbin, R., *Biological sequence analysis : probalistic models of proteins and nucleic acids*. 1998, Cambridge, UK New York: Cambridge University Press. xi, 356 p.

55. Ahmadian, Y., J.W. Pillow, and L. Paninski, *Efficient Markov Chain Monte Carlo Methods for Decoding Neural Spike Trains.* Neural Computation, 2010. **23**(1): p. 46-96.

56. Escola, S., A. Fontanini, D. Katz, and L. Paninski, *Hidden Markov Models for the Stimulus-Response Relationships of Multistate Neural Systems.* Neural Computation, 2011. **23**(5): p. 1071-1132.

57. Koller, D. and N. Friedman, *Probabilistic graphical models : principles and techniques*. Adaptive computation and machine learning. 2009, Cambridge, MA: MIT Press. xxi, 1231 p.

58. Griliches, Z. and M.D. Intriligator, *Handbook of econometrics*. Handbooks in economics. Vol. 1. 1983, New York, N.Y.: Elsevier Science Pub. Co.

59. Husmeier, D., R. Dybowski, and S. Roberts, *Probabilistic modeling in bioinformatics and medical informatics*. Advanced information and knowledge processing,. 2005, London: Springer. xix, 504 p.

60. Rasnik, I., S.A. McKinney, and T. Ha, *Nonblinking and long-lasting single-molecule fluorescence imaging.* Nature Methods, 2006. **3**(11): p. 891-893.

61. Zheng, Q., M.F. Juette, S. Jockusch, M.R. Wasserman, Z. Zhou, R.B. Altman, and S.C. Blanchard, *Ultra-stable organic fluorophores for single-molecule research.* Chemical Society Reviews, 2014. **43**(4): p. 1044-1056.

62. Altman, R.B., Q. Zheng, Z. Zhou, D.S. Terry, J.D. Warren, and S.C. Blanchard, *E nhanced photostability of cyanine fluorophores across the visible spectrum.* Nat Meth, 2012. **9**(5): p. 428-429.

63. Zheng, Q., S. Jockusch, Z. Zhou, R.B. Altman, J.D. Warren, N.J. Turro, and S.C. Blanchard, *On the Mechanisms of Cyanine Fluorophore Photostabilization.* The J ournal of Physical Chemistry Letters, 2012. **3**(16): p. 2200-2203.

64. Vosch, T., Y. Antoku, J.-C. Hsiang, C.I. Richards, J.I. Gonzalez, and R.M. Dickso n, *Strongly emissive individual DNA-encapsulated Ag nanoclusters as single-mole cule fluorophores.* Proc. Natl. Acad. Sci. U. S. A., 2007. **104**(31): p. 12616-12621.

65. Sukhanova, A., J. Devy, L. Venteo, H. Kaplan, M. Artemyev, V. Oleinikov, D. Kli nov, M. Pluot, J.H.M. Cohen, and I. Nabiev, *Biocompatible fluorescent nanocryst als for immunolabeling of membrane proteins and cells.* Analytical Biochemistry, 2004. **324**(1): p. 60-67.

66. Levinson, S.E., L.R. Rabiner, and M.M. Sondhi, *An introduction to the applicatio n of the theory of probabilistic functions of a Markov process to automatic speech recognition.* The Bell System Technical Journal, 1983. **62**(4): p. 1035-1074.

67. Yang, L., B.K. Widjaja, and R. Prasad, *Application of hidden Markov models for s ignature verification.* Pattern Recognition, 1995. **28**(2): p. 161-170.

68. Watkins, L.P. and H. Yang, *Detection of Intensity Change Points in Time-Resolved Single-Molecule Measurements.* The Journal of Physical Chemistry B, 2005. **109**(1): p. 617-628.

69. Hajdziona, M. and A. Molski, *Maximum likelihood-based analysis of single-mole cule photon arrival trajectories.* The Journal of Chemical Physics, 2011. **134**(5): p . 054112.

70. Harte, D.S., *Package "HiddenMarkov"*. 2012, Comprehensive R Archive Network (CRAN).

CHAPTER 4

FITTING EXPERIMENTAL DATA

4.1 Introduction

A promising fluorescent probe should show good photostability, brightness, and spectral purity. Silver nanodots were developed to fulfill those characteristics using oligonucleotide scaffolds [1-3]. Synthesized silver nanodots showed long photostability (~10 min), and their excitation and emission wavelength could be tuned by optimizing the DNA sequence.

Silver nanodots are optically modulatable by longer-wavelength secondary laser, in which molecules at dark state are depopulated toward singlet transition manifolds by reverse intersystem crossing [4, 5]. Patel *et. al.* proposed that dark states are generated from photoinduced charge transfer from silver nanoclusters to the DNA scaffold [6]. Three relaxation pathways of photoexcited silver nanodots were proposed, including radiative decay, nonradiative decay, and charge transfer. By transient absorption spectroscopy, decay times of radiative and nonradiative decay were measured to have ~2 ns and ~500 fs, respectively. It was proposed that dark state was generated by charge transfer to the encapsulating oligonucleotide. During charge transfer state, silver nanodots cannot emit photons; thus it is called the dark state. Decay times of the long-lived dark state were extracted from FCS of photon time traces collected from three different species of silver nanodots [6]. These decay times were observed to be 10~30 μs. Illuminating the secondary laser reversibly increased fluorescence intensity via reverse intersystem crossing, which recovers emissive manifolds by reducing the population of molecules in the dark state [4, 5]. The ground state recovery rate from the dark state increases when the secondary laser illuminates the nanodot, thereby enhancing

fluorescence intensity. Background emission from heterogeneous environments does not respond at the secondary excitation wavelength. Therefore we can selectively enhance emission signal from silver nanodots [7].

In this chapter, time trace of fluorescence photons from silver nanodot are to be analyzed using PbPHMM. Photophysical parameters of silver nanodots will be separately estimated according to illumination pattern.

4.2 Experimental

4.2.1 Synthesis of silver nanodot

Silver nanodots were synthesized by reducing silver nitrate ($AgNO_3$, Sigma-Aldrich, 99.9999%) with sodium borohydride ($NaBH_4$, Sigma-Aldrich, 98%) using single-stranded DNA templates. The sequence of the oligonucleotide (Integrated DNA Technologies) was 5'-CCCTAACTCCCC-3', and it was dissolved into deionized water (~17.1 MΩ). The molar ratio of $AgNO_3$, $NaBH_4$, and oligonucleotide was kept to 300:150:50 (/nmol), and they were mixed with 900 μl of deionized water. After being refrigerated for 5~6 hours, the Ag-DNA nanodot solution was mixed with saturated PVA solution with volumetric ratio of 1:1000. The mixture was spin coated on microscope cover glass for 1 min.

4.2.2 Single molecule measurement

A schematic of experimental setup for optical modulation is shown in Figure 4.1. All measurements were done on an inverted optical microscope (Olympus, IX-71) with 60x oil objective (Olympus). Focusing was stabilized by automatic focus drift correcting system (MCL C-Focus). Silver nanodots were excited by He-Ne laser at 633 nm (primary

excitation) and optically modulated by Tunable Ti-sapphire laser (Coherent) at 800 nm (secondary excitation). The secondary Ti-sapphire laser was manually controlled by a beam shutter (Thorlabs, SH05). Excitation light was blocked by a combination of optical filters and dichroic mirrors. Photons, passed through the side port of the microscope, were guided to two avalanche photodiodes (APD, Perkin Elmer, SPCM-AQR-15-FC) by optical fibers.

Signal from one APD was artificially delayed by 1.5 μs with a digital delay generator (Stanford Research Systems, DG645) for better time resolution while the signal from the other APD was not changed. Using a routing module (Becker & Hickl GmbH, HRT-41), signals from two different APDs were combined into a single time trace, which was collected by time-correlated single photon counting modules (Becker & Hickl GmbH, SPC-630). Photon streams from immobilized AcGFP molecules were collected with one APD.

EMCCD (Andor Technology) camera was used to monitor the optical alignment of primary and secondary laser spot and find fluorescent silver nanodots. Using a scanning stage and sub-nanometer controller (MCL, Nano-Drive), sample was positioned on an observation point where photons were collected with maximum efficiency. Focusing was stabilized by automatic focus drift correcting system (MCL, C-Focus).

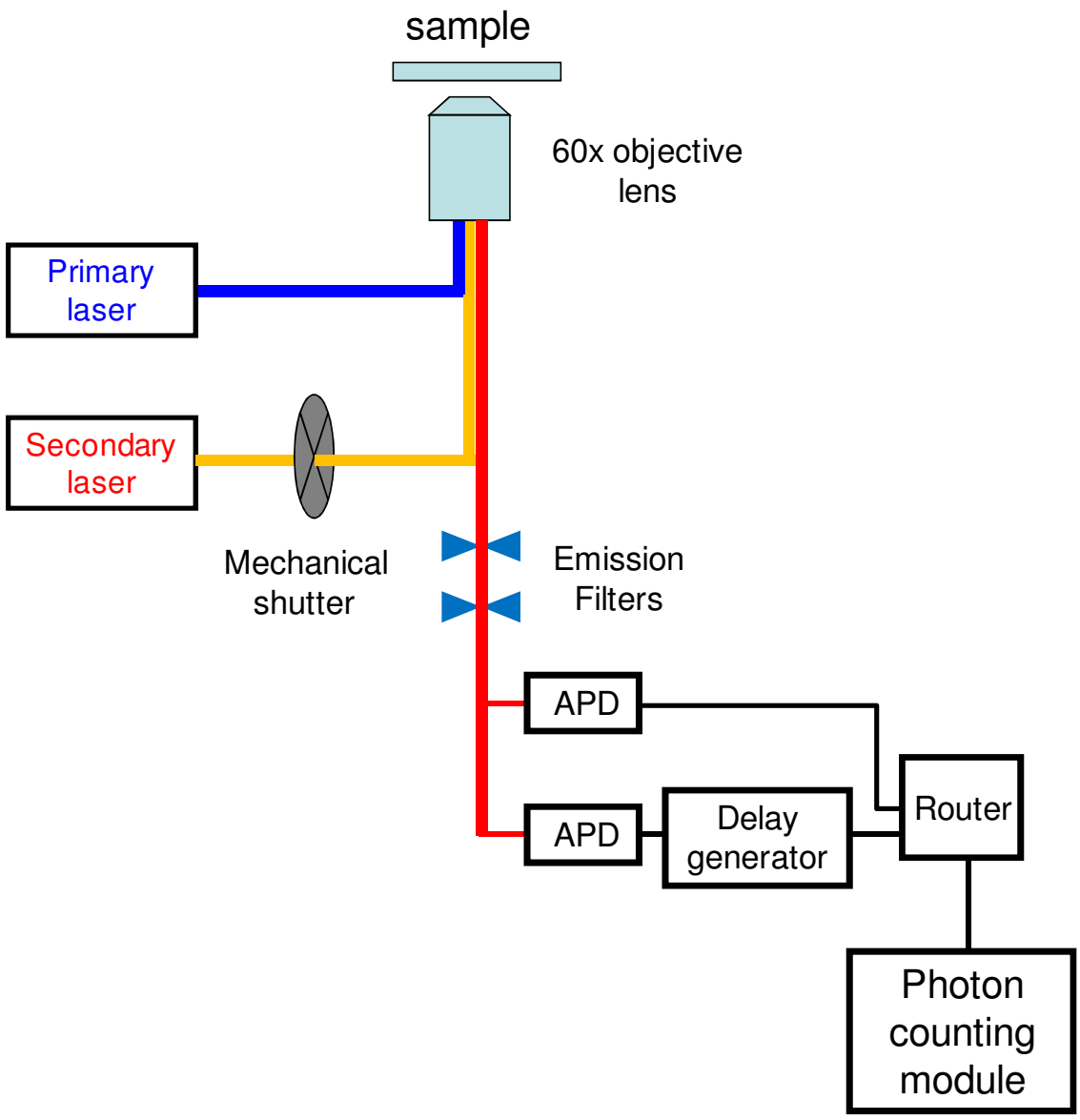

Figure 4.1 The diagram of experimental setup for optical modulation of silver nanodots.

4.3 Results and Discussion

4.3.1 AgDNA

Synthesized silver nanodot with DNA-scaffold excited at 633 nm showed fluorescent emission at 700nm with high photostability, showing photobleaching quantum yield around 5×10^{-7}. (Figure 4.2(a)). In average, 5×10^{5} photons were collcted from single individual silver nanodot molecule before photobleaching. Considering the detection efficiency (0.03) and fluorescence quantum yield (0.35), the number of detected photons were converted to photobleaching quantum yield around 5×10^{-7}. A 50 nM Silver nanodot sample, immobilized in saturated PVA solution, was excited by defocused light at 633nm to find a fluorescent spot as shown in Figure 4.2(b). After finding a bright spot, it was positioned to the point for maximum photon collection using a piezoelectric nanopositioning stage.

Photon time traces of optically modulated silver nanodots were collected using the TCSPC setup in Figure 4.1. An example of the time traces is presented in Figure 4.2(b). Black and grey lines denote time traces collected by different channels. Primary and secondary excitation intensity was 1.9kW/cm^2 and 3.5kW/cm^2, respectively. The secondary laser at 805 nm was controlled manually with a beam shutter as shown in the illumination pattern at the bottom of Figure 4.2(c).

Illuminating the secondary laser increased fluorescence intensity via reverse intersystem crossing, which recovers emissive manifolds by reducing the population of molecules in dark state [4, 5]. Escape rate from the dark state increases when the secondary laser is on, enhancing fluorescence intensity. Autocorrelations with single- and dual-laser illumination are shown in Figure 4.2(d). Autocorrelation with dual laser illumination (dashed line) showed smaller contrast than autocorrelation with single laser illumination (dotted line), showing the contrast, τ_{off}/τ_{on}, decreased by the secondary laser

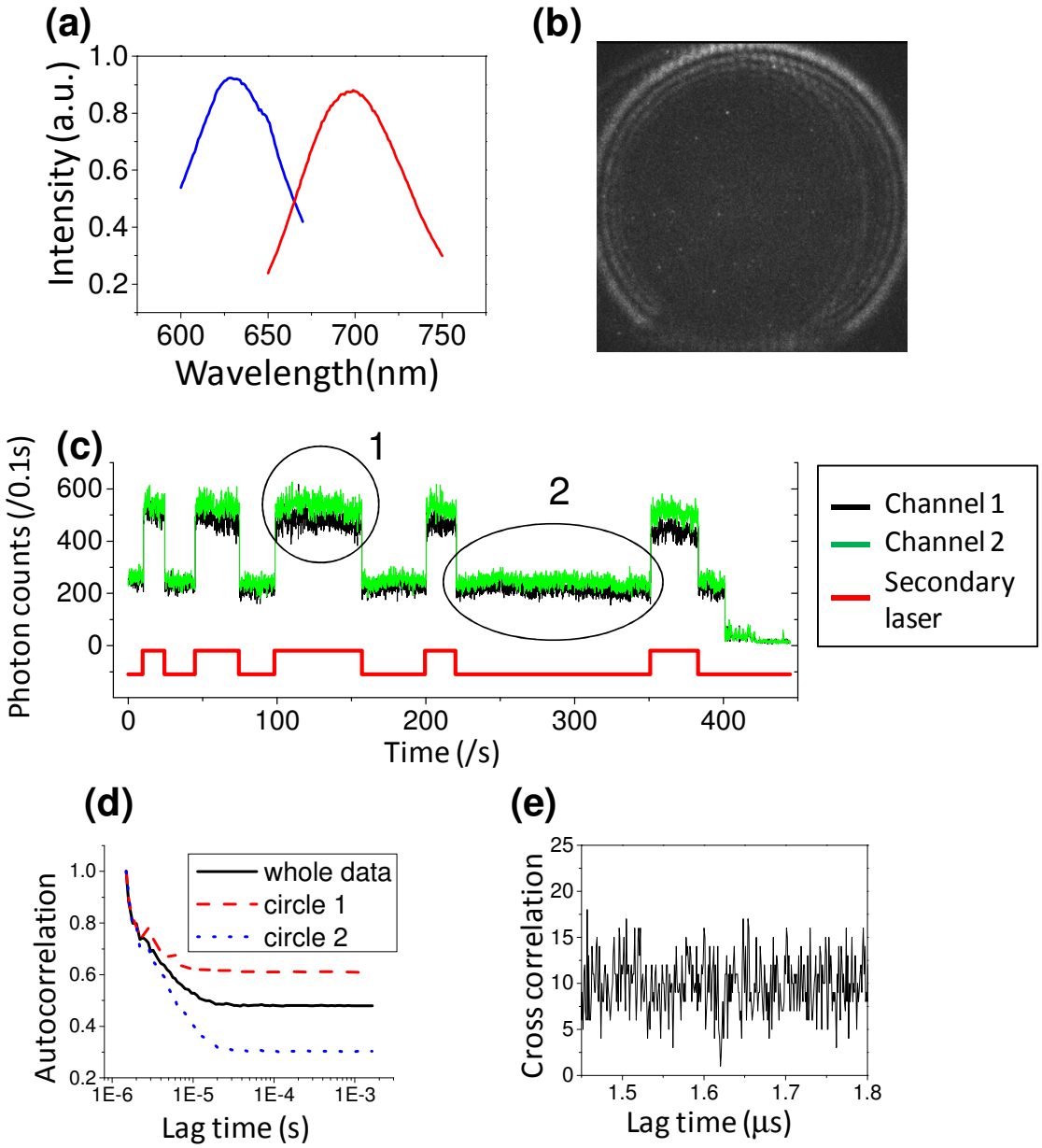

Figure 4.2 Fluorescence signals from typical optically-modulated, single silver nanodots. (a) Excitation and emission spectra of silver nanodots. (b) The defocused image of excited silver nanodot molecules. (c) Time trace of fluorescence intensity measured by two separated channels. Illumination pattern of secondary excitation laser is shown below the time trace. (d) Autocorrelation of whole data and photons collected under one (circle 1 in (c)) and two (circle 2 in (c)) lasers. (e) Cross correlation of two time traces collected by two different detectors.

illumination. On- and off-time, which are the inverse of escape rates from bright and dark manifolds, respectively, were determined by exponential fitting of the autocorrelation curves using eq. (4.1)

$$G(\tau) = A + Be^{-\tau/t_c}$$

$$\frac{1}{t_c} = \frac{1}{\tau_{\text{on}}} + \frac{1}{\tau_{\text{off}}}, \quad \frac{A}{B} = \frac{\tau_{\text{on}}}{\tau_{\text{off}}} \tag{4.1}$$

While τ_{on} did not vary significantly (8.8 μs in circle 1 and 8.4 μs in circle 2), τ_{off} decreased by ~75% from 18 μs to 4.6 μs with secondary laser illumination. It was confirmed that the time trace was from a single emitter by cross correlation. Delay time for signals from the second APD was set to 1.5 μs, and the delay generator has an intrinsic trigger delay around 100 ns. An antibunching dip that appeared at 1.6 μs in second-order cross correlation verified a single emitter (Figure 4.2(e)).

On- and off-time, τ_{on} and τ_{off}, were extracted from the time trace in Figure 4.2(c) using autocorrelation, and PbPHMM (Figure 4.3). The time trace was divided whether the secondary laser was irradiated or not, then the individual traces were analyzed separately by autocorrelation (Figure 4.3(a)) and PbPHMM (Figure 4.3(b)). Instead of estimating τ_{on} directly from the time trace, Φ_{Dark} is extracted by PbPHMM, then τ_{on} is calculated using the relation $\tau_{on} = 1/k_{exc}/ \Phi_{Dark}$. τ_{off} and τ_{on} estimated by autocorrelation and PbPHMM are similar to each other. In average τ_{off} decreased by ~3 with the secondary laser illumination. The ratio of τ_{off} to τ_{on} showed more consistency, which reversibly changed with illumination conditions..

Photophysical model parameters from all collected photon time traces using autocorrelation and PbPHMM were compared in Figure 4.4 and Table 4.1. Figure 4.4 shows BIC values from four photon time traces as a function of assumed number of hidden states. In both conditions with single- and dual-laser illumination, BIC was

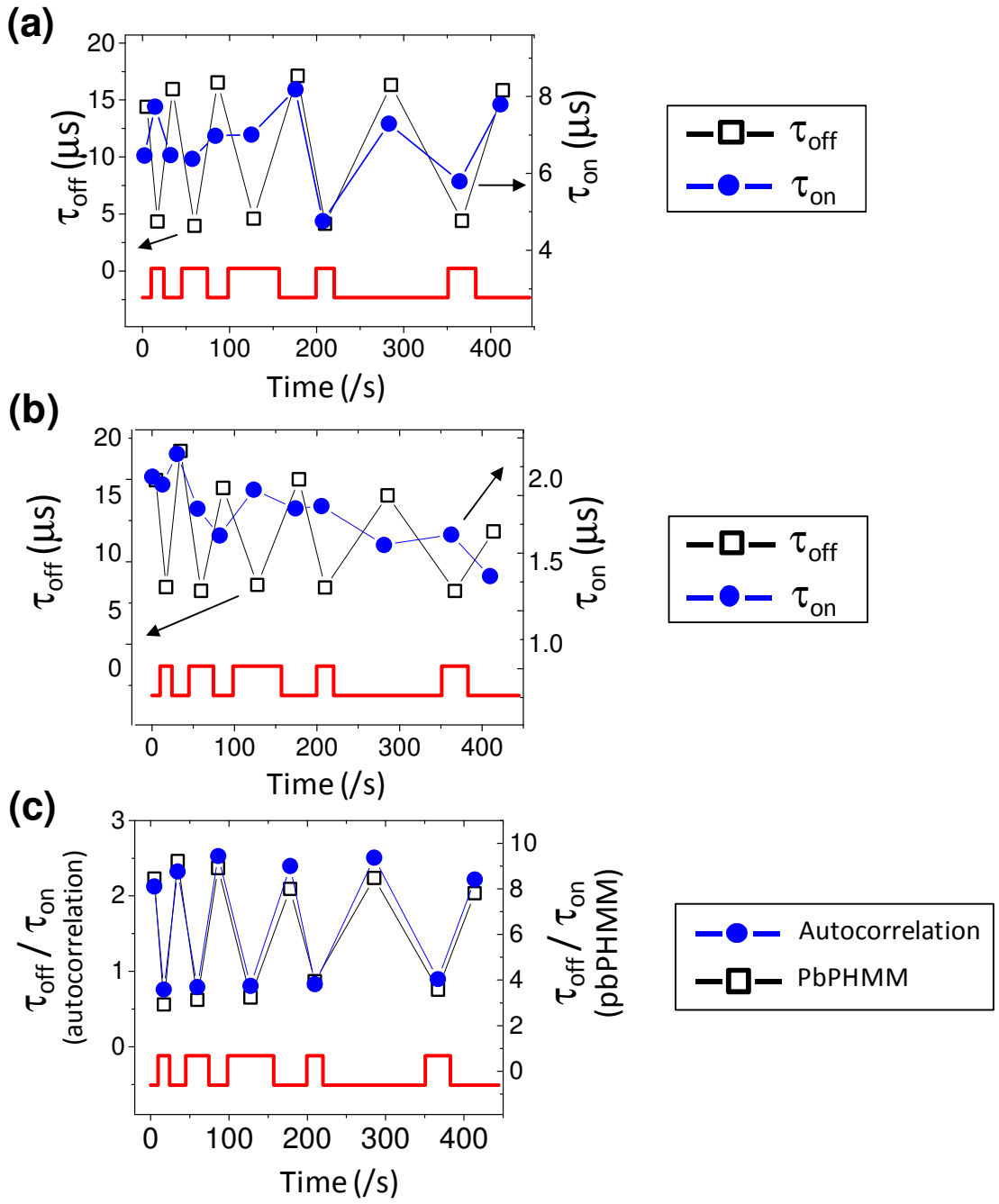

Figure 4.3 Fitting results of photon time trace from optically-modulated, single silver nanodot. (a) τ_{off} (empty square) and τ_{on} (solid circle) estimated by autocorrelation (b) τ_{off} (empty square) and τ_{on} (solid circle) estimated by PbPHMM (c) the ratio of τ_{off} to τ_{on} by autocorrelation (empty square) and PbPHMM (solid circle).

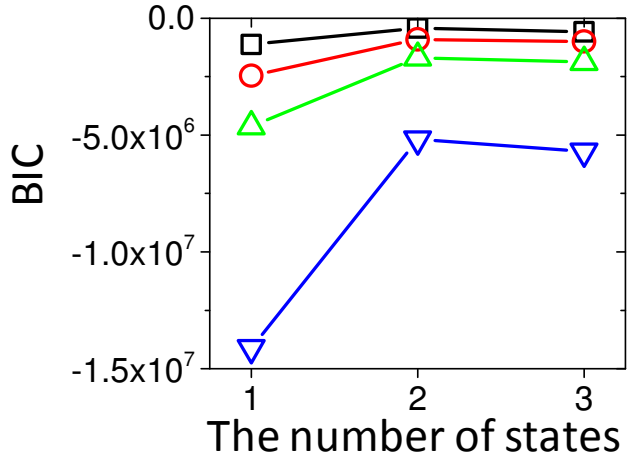

Figure 4.4 BIC as a function of the number of hidden states

Table 4.1 Average values and standard deviations of τ_{off}, τ_{on}, Φ_{Dark}, and τ_{off} / τ_{on} calculated from 45 photon time traces collected from optically modulated silver nanodots according to illumination conditions.

	PbPHMM		**Autocorrelation**	
	Single-laser	Dual-laser	Single-laser	Dual -laser
τ_{off} (μs)	13.6 (±3.6)	8.33(±2.9)	14.9 (±3.8)	5.32 (±2.00)
τ_{on} (μs)	8.95 (±1.15)	7.79 (±6.58)	10.9 (±5.8)	14.3 (±10.7)
Φ_{Dark}	0.0140 (±0.0108)	0.0161 (±0.0136)	0.0144 (±0.0063)	0.0138 (±0.0087)
τ_{off}/τ_{on}	1.52(±0.45)	1.07 (±0.97)	1.36 (±0.81)	0.361 (±306)

maximized at two hidden states, which means one dark and one bright state exist in a photophysical model of fluorescent silver nanodots regardless of illumination conditions. Some of photon time traces could not be analyzed by PbPHMM because transition and emission matrices assuming more than two hidden states were not photophysically relevant.

4.4 Conclusion

PbPHMM was applied to the photophysical model building from single molecule photon time traces from optically-modulated silver nanodots. BIC was used to determine the most probable number of hidden states. One bright and one dark state were predicted from photon trajectories from silver nanodots whether the secondary laser is on or off. PbPHMM and autocorrelation returned similar on- and off-time. The secondary laser illumination shortened τ_{off} three times, and the contrast τ_{off}/τ_{on} reversibly was modulated by illumination conditions.

4.5 References

1. Petty, J.T., J. Zheng, N.V. Hud, and R.M. Dickson, *DNA-Templated Ag Nanoclust er Formation.* Journal of the American Chemical Society, 2004. **126**(16): p. 5207-5212.

2. Vosch, T., Y. Antoku, J.-C. Hsiang, C.I. Richards, J.I. Gonzalez, and R.M. Dickso n, *Strongly emissive individual DNA-encapsulated Ag nanoclusters as single-mole cule fluorophores.* Proceedings of the National Academy of Sciences, 2007. **104**(3 1): p. 12616-12621.

3. Richards, C.I., S. Choi, J.-C. Hsiang, Y. Antoku, T. Vosch, A. Bongiorno, Y.-L. Tz eng, and R.M. Dickson, *Oligonucleotide-Stabilized Ag Nanocluster Fluorophores.* Journal of the American Chemical Society, 2008. **130**(15): p. 5038-5039.

4. Ringemann, C., A. Schönle, A. Giske, C.v. Middendorff, S.W. Hell, and C. Eggeli ng, *Enhancing Fluorescence Brightness: Effect of Reverse Intersystem Crossing S tudied by Fluorescence Fluctuation Spectroscopy.* ChemPhysChem, 2008. **9**(4): p. 612-624.

5. Richards, C.I., J.-C. Hsiang, D. Senapati, S. Patel, J. Yu, T. Vosch, and R.M. Dicks on, *Optically Modulated Fluorophores for Selective Fluorescence Signal Recover y.* Journal of the American Chemical Society, 2009. **131**(13): p. 4619-4621.

6. Patel, S.A., M. Cozzuol, J.M. Hales, C.I. Richards, M. Sartin, J.-C. Hsiang, T. Vos ch, J.W. Perry, and R.M. Dickson, *Electron Transfer-Induced Blinking in Ag Nano dot Fluorescence.* J. Phys. Chem. C, 2009. **113**(47): p. 20264-20270.

7. Richards, C.I., J.-C. Hsiang, and R.M. Dickson, *Synchronously Amplified Fluores cence Image Recovery (SAFIRe).* The Journal of Physical Chemistry B, 2009. **114** (1): p. 660-665.

CHAPTER 5

CONCLUSIONS AND FUTURE OUTLOOK

Optically-controllable fluorescence emission led to more sensitive biological imaging with enhanced resolution. To extend and improve imaging quality, it is crucial to understand which photophysical and photochemical processes occur. The objective of this thesis is to develop a methodology in order to build a photophysical model from photon time traces, to explain the connectivity of underlying states, and to estimate photophysical parameters of the underlying model.

A HMM is a double-stochastic model which can reveal hidden dynamics in a robust and unbiased way. First, Poisson-distributed time binned photon trajectories were analyzed using HMM. The BW algorithm was modified by introducing Poisson modification and an additional penalizing term, localization error. Photophysically-irrelevant training of the emission matrix was removed by restraining the emission matrix to be Poissonian. Over-fitting problems were solved by localization error, thereby determining the simplest model which efficiently describes the time-binned fluorescence trajectories. The improved model selection accuracy was maintained even if the intensity trajectory is very short.

In this thesis, PbPHMM has been developed to build a photophysical model from photon time traces, overcoming following problems in time binned analysis. One limitation in time-binned analysis is that underlying dynamics may be lost if the binning time is longer than the time scale of the dynamics. Additionally, it may be extremely

complicated to extract the connectivity and transition rates between hidden states due to the lack of closed-form solution of master equation for complex model with more than one dark state. Analysis of photon detection rate assuming Poisson processes can be a good solution to solve these problems. Realistic photon time traces, however, often exhibit non-Poisson behavior due to non-unity detection efficiency and background noise. The solution of the master equation is not required because every photon waiting time is analyzed instead of the probability of photon detection within a given bin time.

The model building process starts with the initial values of the number of dark states, their quantum yield, and lifetime. The most probable number of dark states is determined when the information criteria are maximized. In photophysical model building, every photon is assigned to newly-defined states, which are different from photophysical states. As shown in Figure 3.2, a state *n* photon is detected after transitions to $Dark_{n-1}$. Photon time traces were generated on various conditions, based on the 3-state model. PbPHMM demonstrated reliable fitting performances in determining the number of dark states, estimating photophysical parameters including dark state quantum yield, and lifetime.

PbPHMM was applied to the photophysical model building from single molecule photon time traces, collected from optically-modulated silver nanodots. The information criteria were maximized at two states; whether the secondary laser is on or off, showing that the secondary laser illumination did not change the number of underlying photophysical states. The lifetime of dark state was shortened by secondary laser illumination. Since the underlying model is determined by PbPHMM, autocorrelation function analysis can be used to calculate on- and off-time. The on- and off-time can be

calculated by exponential fitting of autocorrelation function only if the underlying model is known.

There are several areas that can be improved in PbPHMM. In evaluating parameter estimation, the lowest root-mean-square value of relative error on typical experimental conditions was ~25%. The best estimation was achieved by averaging transition and emission matrices over 50 simulated data traces. However, the method could not be applied to analyze experimental data because experimental conditions during multiple data collection were not consistent. Random orientation of dipole moment within silver nanodot molecules can to be considered as a method to calculate excitation rate of single molecules more precisely. Photophysical parameters will be estimated more accurately if experimental conditions including excitation intensity and detection efficiency are calibrated. A few things in fitting codes remain to be further improved. Theoretically PbPHMM has unlimited time resolution. However the emission matrix is written as a sparse array, and rounded photon waiting times are saved into discrete elements in the array instead of dealing with raw photon waiting time.

In this study, the most probable model is determined among models with different number of dark states. With a given model, it was assumed that a photon is detected only after excitation followed by radiative decay. The other possible process of photon emission involves excitation, dark state transition, recovery into singlet manifolds via reverse intersystem crossing, and radiative decay. Photon waiting time containing these processes was not considered in selecting a photophysical model in this thesis. Another possibility is branched dark state decay, in which the molecule is relaxed to ground photophysical state via consecutive dark state relaxations. PbPHMM will determine the

most probable model by comparing information criteria, and will allow the estimation of photophysical parameters from transition and emission matrices, which can be derived from corresponding models in the same way presented in this thesis. Fluorescence intermittency of semiconductor quantum dots is reported to follow power-law distribution. The underlying model of fluorescence of quantum dots can be extracted if the power-law distribution is introduced to derive the emission matrix.

Combined with other techniques such as FCS, PbPHMM will open new perspectives for explaining molecular processes in dynamic samples, by offering the underlying physical models.

APPENDIX A

MASTER EQUATION

Third-order joint probability of stochastic variables x_1, x_2, x_3 at time t_1, t_2, t_3, when $t_1 < t_2 < t_3$ is;

$$\mathrm{P}(x_1,t_1;x_2,t_2;x_3,t_3) = \mathrm{P}(x_3,t_3|x_1,t_1;x_2,t_2)\mathrm{P}(x_1,t_1;x_2,t_2)$$

$$= \mathrm{P}(x_3,t_3|x_1,t_1;x_2,t_2)\mathrm{P}(x_2,t_2|x_1,t_1)\mathrm{P}(x_1,t_1) \quad \text{(A.1)}$$

If the stochastic process has Markov property, current state only depends on previous state.

$$\mathrm{P}(x_1,t_1;x_2,t_2;x_3,t_3) = \mathrm{P}(x_3,t_3|x_2,t_2)\mathrm{P}(x_2,t_2|x_1,t_1)\mathrm{P}(x_1,t_1) \quad \text{(A.2)}$$

Integrate over x_2 then modified to

$$\mathrm{P}(x_3,t_3 \mid x_1,t_1) = \int \mathrm{P}(x_3,t_3 \mid x_2,t_2)\mathrm{P}(x_2,t_2 \mid x_1,t_1)\, dx_2 \quad \text{(A.3)}$$

Eq. (A.3) is called the Chapman-Kolmogorov equation.

The short-time behavior of the transition probability from x'' at time t to x at time $t + \tau$ can be written as series expansion with respect to the time interval t as follows

$$\mathrm{P}(x,t+\tau \mid x'',t) = [1 - \tau \int w_t(x"|x)\, dx"]\delta(x - \mathrm{x}") + \tau\, w_t(x|x") + O(\tau^2) \quad \text{(A.4)}$$

where $\int w_t(x"|x)\, dx"$ is the transition rate per unit time from x to x'' at time t. The first term in the series expansion is the probability that no transition occurs during time interval t. The second term is the probability that the transition from x to x'' occurs. The third term $O(\tau^2)$ goes to zero when $\tau^2 \to 0$.

From (A.3) and (A.4) together [1];

$$\mathrm{P}(x,t+\tau \mid x',t') = \int \mathrm{P}(x,t+\tau \mid x",t)\mathrm{P}(x",t \mid x',t')\, dx"$$

$$= \int \left[1 - \tau \int w_t(x''|x)\, dx''\right] \delta(x - x'')\mathrm{P}(x", t \mid x', t')\, dx"$$

$$+ \int \tau\, w_t(x|x")\mathrm{P}(x", t \mid x', t')\, dx" + \int O(\tau^2)\mathrm{P}(x", t \mid x', t')\, dx" \tag{A.5}$$

$$[\mathrm{P}(x, t + \tau \mid x', t') - \mathrm{P}(x, t \mid x', t')]/\tau$$

$$= \int w_t(x|x")\mathrm{P}(x", t \mid x', t')\, dx" - \int w_t(x''|x)\mathrm{P}(x, t \mid x', t')\, dx'' \tag{A.6}$$

Taking limit t→0, then

$$\frac{\partial}{\partial t}\mathrm{P}(x, t \mid x', t') = \int w_t(x|x")\mathrm{P}(x", t \mid x', t')\, dx" - \int w_t(x''|x)\mathrm{P}(x, t \mid x', t')\, dx'' \tag{A.7}$$

Multiplying by $\mathrm{P}(x', t')$, integrating over x', and replacing $x"$ by x', then we get

$$\frac{\partial}{\partial t}\mathrm{P}(x, t) = \int w_t(x|x')\mathrm{P}(x', t)\, dx' - \int w_t(x'|x)\mathrm{P}(x, t)\, dx' \tag{A.8}$$

Eq. (A.8) is called the master equation. Assuming a time homogeneous process in discrete space, Eq. (A.8) can be written as follows;

$$\frac{\partial}{\partial t}\mathrm{P}(x, t) = \sum_{x \neq x'}[w(x|x')\mathrm{P}(x', t) - w(x'|x)\mathrm{P}(x, t)] \tag{A.9}$$

Convert Eq. (A.8) into matrix form

$$\frac{\partial}{\partial t}\mathrm{P}(x, \Delta t \mid x', 0) = \boldsymbol{K}\mathrm{P}(x, \Delta t \mid x', 0) \tag{A.10}$$

The master equation for two-state model can be derived using Eq. (A.10) and (A.11).

$$\text{on} \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} \text{off} \qquad G(\tau) = A + Be^{-\tau/t_c}$$

$$\frac{1}{t_c} = \frac{1}{\tau_{\text{on}}} + \frac{1}{\tau_{\text{off}}}, \quad \frac{A}{B} = \frac{\tau_{\text{on}}}{\tau_{\text{off}}} \qquad \boldsymbol{K} = \begin{pmatrix} -k_{\text{on}} & k_{\text{off}} \\ k_{\text{on}} & -k_{\text{off}} \end{pmatrix} \tag{A.11}$$

With initial condition P(on,0)=1,

$$\mathrm{P}(on, \tau \,|on, 0) = \frac{k_{\text{off}}}{k_{\text{off}} + k_{\text{on}}} + \frac{k_{\text{on}}}{k_{\text{off}} + k_{\text{on}}} \mathrm{e}^{-(k_{\text{off}} + k_{\text{on}})\tau}$$

Fluorescence intensity is proportional to the probability of bright state. Therefore, autocorrelation function of fluorescence intensity is proportional to the joint probability of bright state after delay time τ given bright state at time zero as follows;

$$\mathrm{G}(\tau) = \mathrm{A} + \mathrm{B}\mathrm{e}^{-\tau/t_c}, \qquad \frac{1}{t_c} = \frac{1}{\tau_{\text{on}}} + \frac{1}{\tau_{\text{off}}}, \qquad \frac{\mathrm{A}}{\mathrm{B}} = \frac{\tau_{\text{on}}}{\tau_{\text{off}}}$$

,where $\tau_{\text{on}} = \frac{1}{k_{\text{on}}}$ and $\tau_{\text{off}} = \frac{1}{k_{\text{off}}}$.

Reference

1. Mahnke, R., J. Kaupuzs, and I.O. Lubashevskyi, *Physics of stochastic processes : how randomness acts in time*. Physics textbook. 2009, Weinheim, Chichester: Wil ey-VCH ;John Wiley distributor. xvii, 430 p.

APPENDIX B

DERIVING TRANSITION MATRIX

Deriving Eq. (3.20) starts from counting the number of transitions into the allowed photophysical processes. For example, photophysical processes before a photon at state 2 is detected are divided into two stages as shown in Eq. (3.15) and Figure 3.3.

As shown in Figure 3.3, Dark_1 is accessed once after stage 1, and may be accessed multiple times before a state 2 photon is detected. The sum of accessed number of Dark_1 before state 2 photon is detected, $N(\text{Dark}_1, \text{state 2})$, is as follows:

$$N(\text{Dark}_1, \text{state 2}) = 1 + \frac{\Phi_{\text{Dark1}}}{1-p_2} = 1 + \frac{\Phi_{\text{Dark1}}}{\Phi_{\text{FL}}\Phi_{\text{Det}}+\Phi_{\text{Dark2}}} \quad \text{(B.1)}$$

In the same way, the accessed number of Dark_2 before state 3 photon is detected, $N(\text{Dark}_2, \text{state 3})$, can be written as follows:

$$N(\text{Dark}_2, \text{state 3}) = 1 + \frac{\Phi_{\text{Dark2}}}{\Phi_{\text{FL}}\Phi_{\text{Det}}} \quad \text{(B.2)}$$

After a fluorescence photon is detected, the molecule always resides in ground photophysical state, and then will repeat excitation/de-excitation cycles until the next photon is detected. Therefore, photon detections are divided by ground photophysical state residence, and the state of current photon is independent upon the state of previous photon when noise level is zero. In this case, the independent state probability can be considered a stationary state probability, which is the same as the population ratio of each state [1]. The population ratio of photons at each state is derived counting the accessed number of dark state before a photon is detected.

If a sufficient number of photon is detected, the ratio of accessed number of each photophysical processes will be the same as the ratio of quantum yields of each processes.

Additionally, $Dark_2$ is only accessed during photon waiting times of state 3 photons. Therefore, the number of state 3 photons will be as follows:

$$N(\text{state 3}) = \frac{\text{the accessed number of } Dark_2 \text{ during collecting a whole time trace}}{\text{the accessed number of } Dark_2 \text{before a state 3 photon is detected}}$$

$$= N_{total} \frac{\Phi_{FL}\Phi_{Det}\Phi_{Dark2}}{\Phi_{FL}\Phi_{Det}+\Phi_{Dark2}} \quad \text{(B.3)}$$

where N_{total} is the number of photophysical state transitions during photon collection.

In the same way, the number of state 2 photons can be estimate by counting the accessed number to $Dark_1$. It should be noted that $Dark_1$ is allowed both during state 2 and state 3. Therefore, the number of state 2 photons will be as follows:

$$N(\text{state 2}) = \frac{(N_{total}\Phi_{Dark1} - \text{the accessed number of } Dark_1 \text{during state 3})}{\text{the accessed number of } Dark_1 \text{before a state 2 photon is detected}}$$

$$= N_{total} \frac{(\Phi_{FL}\Phi_{Det})^2\Phi_{Dark1}}{(\Phi_{FL}\Phi_{Det}+\Phi_{Dark2})(\Phi_{FL}\Phi_{Det}+\Phi_{Dark1}+\Phi_{Dark2})} \quad \text{(B.4)}$$

The number of state 1 photons can be estimate by counting missed photons of which quantum yield is $\Phi_{FL}(1- \Phi_{Det})$. Photon missing in all states should be considered.

$$N(\text{state 1}) = \frac{N_{total}\Phi_{FL}(1-\Phi_{Det}) - \text{the number of missed photon during state 3 and state 2}}{\text{the number of missed photon during state 1}}$$

$$= N_{total} \frac{(\Phi_{FL}\Phi_{Det})^2}{\Phi_{FL}\Phi_{Det}+\Phi_{Dark1}+\Phi_{Dark2}} \quad \text{(B.5)}$$

The transition probabilities of state 1~ state 3 are the ratio of population of each state as follows:

$$P(\text{state 1}|\text{state x}) = \frac{\Phi_{FL}\Phi_{Det}}{\Phi_{FL}\Phi_{Det} + \Phi_{Dark1} + \Phi_{Dark2}}$$

$$P(\text{state 2}|\text{state x}) = \frac{\Phi_{FL}\Phi_{Det}\Phi_{Dark1}}{(\Phi_{FL}\Phi_{Det} + \Phi_{Dark2})(\Phi_{FL}\Phi_{Det} + \Phi_{Dark1} + \Phi_{Dark2})}$$

$$P(\text{state 3}|\text{state x}) = \frac{\Phi_{Dark2}}{\Phi_{FL}\Phi_{Det} + \Phi_{Dark2}}$$

References

1. Fraser, A.M., *Hidden Markov models and dynamical systems*. 2008, Philadelphia, PA: Society for Industrial and Applied Mathematics. xii, 132 p.

2. Dwyer, M.G., N. Bergsland, E. Saluste, J. Sharma, Z. Jaisani, J. Durfee, N. Abdelr ahman, A. Minagar, R. Hoque, F.E. Munschauer, 3rd, and R. Zivadinov, *Applicati on of hidden Markov random field approach for quantification of perfusion/diffusi on mismatch in acute ischemic stroke.* Neurol Res, 2008. **30**(Copyright (C) 2012 U.S. National Library of Medicine.): p. 827-834.

APPENDIX C

PBPHMM CODES

```
function [k_exc1,k_exc2,t_ground]= absc(input_param)
% Calculate excitation rate and dwell time at ground state before
% excitation
% k_exc1 : excitation rate only considering singlet transition. limited
by
% fluorescent decay or internal conversion
%
% k_exc2 : overall excitation rate limited by decay time including
% fluorescent decay, internal conversion, and dark state transition.
%
% t_ground : mean dwell time at ground state prior to excitation.
% calculated from the number of absorbed photon by a molecule per unit
% time.
% input_param : a matrix for input photophysical parameters

% The order of parameters in the input parameter matrix
% 1. I_prim : Primary excitation intensity (W/cm2)
% 2. e : absorption coefficient (/(M*cm))
% 3. wavelength (nm)
% 4. F_fl : fluorescence quantum yield
% 5. F_det : detection efficiency
% 6. t_fl : "measured" fluorescence lifetimea
% 7. t_dark1 : shorter dark state lifetime (T1_1 -> S0)
% 8. Shorter-dark state quantum yield : k_ISC1(intersystem crossing
rate)/k_rad
% 9. ratio of k_ReISC (reverse intersystem crossing rate) to k_ISC2
%    (triplet depopulation rate from T1 to S0) : k_ReISC/k_ISC2
%    - Action cross section for reverse intersystem crossing due to
%     secondary laser illumination
% 10. intensity counts of background
% 11. tbin : time step
% 12. collecting time
% 13. t_mod : modulation time (inverse of modulation frequency)
% 14. I_secmax : Secondary excitation intensity for modulation in
W/cm^2.
% 15. wavelength2 : wavelength of secondary laser (W/cm2)
% t_on : average on-time, t_off : average off-time
% state 1 : off-state
% state 2 : on-state
I_prim=input_param(1);
s_abs=input_param(2)*3.82356e-21;
wavelength=input_param(3);
t_fl=input_param(6);
I_sat = 6.6261e-34*2.9979e8/(s_abs*wavelength*10^-9*t_fl);
%Nbins = t_mod/tbin;
Ffl=input_param(4);
Fdark=input_param(8,:);
Fic=1-Ffl-sum(Fdark);
```

```
tdark=input_param(7,:);
t_fl=input_param(6,1);
t_ic=input_param(13,1);

qy=[Ffl,Fic,Fdark];
lifetime=[t_fl,t_ic,tdark];

t_ground=1/(I_prim*s_abs*wavelength*10^-9/(6.6261e-34*2.9979e8));
k_exc1=1/(t_ground + (Ffl*t_fl+Fic*t_ic)/(Ffl+Fic));
k_exc2=1/(t_ground + sum(qy.*lifetime));
```

```
function tacf=acf_fit_2s(acf,start_v,x_unit,x,point1,point2)
% exponential fit autocorrelation curve and return on- and off-time
% acf=[lag time, autocorrelation coefficient]
% x_unit = time unit of autocorrelation
% [t_on, t_off]=acf_fit(acf,x_unit,x,point1,point2)
% start_v = initial point : [initial toff, initial ton]
if nargin == 1
    % acf has both x and y data. x is used in fitting.
    if size(acf,1)<size(acf,2)
        acf=acf';
    end
    x=acf(:,1);
    y=acf(:,2);
    a=acf(1,2)-acf(end,2);
    c=acf(end,2);
elseif nargin == 3
    x=(point1:point2)*x_unit;
    y=acf(point1:point2);
    a=acf(point1)-acf(point2);
    c=acf(point2);
elseif nargin == 4
    point2=length(acf);
    point1=2;
    %x=x(point1:point2);
    y=acf;
    a=acf(1)-acf(end);
    c=acf(end);
elseif nargin == 5
    point2=length(acf);
    y=acf(point1:point2);
    a=acf(point1)-acf(point2);
    c=acf(point2);
end
if size(x,1)<size(x,2)
    x=x';
end
if nargin==1
    %     f=fit(x,y-c,'exp1');
    % find the closest point to c+a/e
    b=x(argmin((y-(c+a/exp(1))).^2));
    decay1=fittype('a*exp(-x/b)+c','dependent','y');
    f=fit(x,y,decay1,'Lower',[0 0 0],'StartPoint',[a b c]);
else
    decay1=fittype('a*exp(-x/b)+c','dependent','y');
%     f=fit(x,y,decay1,'Lower',[0 0
0],'StartPoint',[c*start_v(1)/start_v(2),...
%         1/(1/start_v(1)+1/start_v(2)),c]);
    f=fit(x,y,decay1,'Lower',[0 0 0],'StartPoint',start_v);
end
% a=t_on, b=t_off
% e1=[num2str(fit_acf.m(2)) '=1/a+1/b'];
% e2=['a/b=' num2str(fit_acf.m(3)) '/' num2str(fit_acf.m(1))];
e1=[num2str(1/f.b) '=1/ton+1/toff'];
e2=['ton/toff=' num2str(f.c) '/' num2str(f.a)];
solution_fit=solve(e1,e2,'ton','toff');
t_on=double(solution_fit.ton);t_off=double(solution_fit.toff);
tacf=[t_off,t_on];
```

```
function
[t_acf,fdark,rsq]=acf_fit_3s(acf,start_t,p_param,x_unit,x,point1,point2
)
% Returns on- and off-times at two different time scales
% the fitting formula was derived from master equation for 3-state
model.
% t_acf=[toff1,toff2;Fisc1,Fisc2];
% fdark = dark state quantum yields of two dark states
% 3-level system(one bright, two dark),two exponential decays.
% x_unit = time unit of autocorrelation
% acf = a*exp(-x/b)+c*exp(-x/d)+e
% start_t = [toff1,ton1,toff2,ton2]
% reference
% autocorrelation - 3state2.nb
if nargin < 3
    % acf has both x and y data. x is used in fitting.
    if size(acf,1)<size(acf,2)
        acf=acf';
    end
    x=acf(:,1);
    y=acf(:,2);
    y0=y(1)-y(end);
    yend=acf(end);
elseif nargin < 4
    x=(point1:point2)*x_unit;
    y=acf(point1:point2);
    y0=acf(point1)-acf(point2);
    yend=acf(point2);
elseif nargin<5
    point2=length(acf);
    point1=2;
    %x=x(point1:point2);
    y=acf;
    y0=acf(1)-acf(end);
    yend=acf(end);
elseif nargin<6
    point2=length(acf);
    y=acf(point1:point2);
    y0=acf(point1)-acf(point2);
    yend=acf(point2);
end
if size(x,1)<size(x,2)
    x=x';
end
% start_t = [toff1,ton1,toff2,ton2]
% start_v = initial [a,b,c,d,e] for curve fitting
%
start_v=[y(end)*start_t(1)/start_t(2),1/(1/start_t(1)+1/start_t(2)),...
%
y(end)*start_t(3)/start_t(4),1/(1/start_t(3)+1/start_t(4)),y(end)];
start_v=start_t;
%%
dec2=fittype('a+b1*exp(-x/t1)+b2*exp(-x/t2)','dependent','y');
fo=fitoptions(dec2);
fo.StartPoint=start_v;
fo.Lower=zeros(1,5);
fo.TolFun=1e-7;
```

```
fo.TolX=1e-7;
fo.MaxIter=1000;
[f,gof] = fit( x, y, dec2, fo);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% t1 = shorter time constant %%%
%%% t2 = longer time constant  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a=f.a;
if f.t1<f.t2
    b1=f.b1;
    t1=f.t1;
    b2=f.b2;
    t2=f.t2;
else
    b2=f.b1;
    t2=f.t1;
    b1=f.b2;
    t1=f.t2;
end
% a,b1,t1,b2,t2
% t1<t2, k1>k2
e1=[num2str(1/t1+1/t2),'=koff1+koff2+kon1+kon2'];
e2=[num2str(1/t1-1/t2),'=sqrt((koff1+koff2+kon1+kon2)^2-
4*(koff2*kon1+koff1*koff2+koff1*kon2))'];
e3=[num2str(b1/a),'=k2*(kon1^2+kon2*(k3-
koff1+koff2+kon2)+kon1*(k3+koff1-
koff2+2*kon2))/(koff1*koff2*(k1*(k1+k3)-4*k2))'];
e4=[num2str(b2/a),'=k2*(-kon1^2+kon2*(k3+koff1-koff2-kon2)+kon1*(k3-
koff1+koff2-2*kon2))/(koff1*koff2*(k1*(-k1+k3)+4*k2))'];
e5='k1=kon1+kon2+koff1+koff2,k2=koff1*(koff2+kon2)+koff2*kon1';
e6='k3=sqrt(k1^2-4*k2)';
s=solve(e1,e2,e3,e4,e5,e6,'koff1','koff2','kon1','kon2','k1','k2','k3')
;
toff1=1./double(s.koff1);toff1=toff1(2);
toff2=1./double(s.koff2);toff2=toff2(2);
ton1=1./double(s.kon1);ton1=ton1(2);
ton2=1./double(s.kon2);ton2=ton2(2);
% Fisc1=double(s.kon1)/k_exc;Fisc1=Fisc1(2);
% Fisc2=double(s.kon2)/k_exc;Fisc2=Fisc2(2);
% p_param=[toff1,toff2;Fisc1,Fisc2];
t_acf=[toff1;ton1;toff2;ton2];
rsq=gof.adjrsquare;
%% calculate quantum yields
if nargin == 3
    k_exc=absc(p_param);
    fdark=1/k_exc./[tacf(2),tacf(4)];
end
%%
% dec2=fittype('a1*exp(-x/t1)+a2*exp(-x/t2)+b','dependent','y');
% start_v2=[start_v(1),start_v(3),start_v(5),start_v(2),start_v(4)];
% [f1,gof1] = fit( x, y, dec2, 'StartPoint', start_v2,'Lower',[0 0 0 0
0]);
% [f2,gof2] = fit( x, y, dec2, 'StartPoint', start_v2);
% if gof1.adjrsquare>gof2.adjrsquare
%     f=f1;
%     gof=gof1;
% else
```

```
%     f=f2;
%     gof=gof2;
% end
% % [f,gof] = fit( x, y, dec2, 'StartPoint', [(y0-yend)/2, 1/1.64e5,
(y0-yend)/2, 1/590, yend] );
% % fit_acf = ezfit(x,y,['y=a*exp(-b*x)+c*exp(-
d*x)+e;a=',num2str(f.a),';b=',...
% %     num2str(-f.b),';c=',num2str(f.c),';d=',num2str(-f.d)]);
% % a=t_on1, b=t_off1, c=t_on2, d=t_off2
% e1=[num2str(1/f.t1) '=1/t_on1+1/t_off1'];
% e2=[num2str(1/f.t2) '=1/t_on2+1/t_off2'];
% e3=['t_on1/t_off1=' num2str(f.b) '/' num2str(f.a1)];
% e4=['t_on2/t_off2=' num2str(f.b) '/' num2str(f.a2)];
% solution_fit=solve(e1,e2,e3,e4,'t_on1','t_off1','t_on2','t_off2');
% % t_acf=[toff1,ton1,toff2,ton2]
%
t_acf=[double(solution_fit.t_off1),double(solution_fit.t_on1),double(so
lution_fit.t_off2),...
%     double(solution_fit.t_on2)];
```

```
function [t_acf,rsq]=acf_fit_3s_f3(acf,start_t,x_unit,x,point1,point2)
% Returns on- and off-times at two different time scales
% F is the fraction of molecules in steady-state dark state.
% autocorrelation function = (1+ F1/(1-F1)*exp(-t/t1))*(1+ F2/(1-
F2)*exp(-t/t2))
% 1/tc=1/ton+1/toff, ton/toff=(1-F)/F
%
% t_acf=[toff1,toff2;Fisc1,Fisc2];
% 3-level system(one bright, two dark),two exponential decays.
% x_unit = time unit of autocorrelation
% start_t = [toff1,ton1,toff2,ton2]
% references
% 1. Patel, Sandeep A. Photophysics of fluorescent silver nanoclusters.
2009.
% dissertation, Georgia Institute of Technology
% 2. Schwille, P., S. Kummer, A.A. Heikal, W.E. Moerner, and W.W. Webb,
% Proceedings of the National Academy of Sciences, 2000. 97(1): p. 151-
156.
%

if nargin < 3
    % acf has both x and y data. x is used in fitting.
    if size(acf,1)<size(acf,2)
        acf=acf';
    end
    x=acf(:,1);
    y=acf(:,2);
    y0=y(1)-y(end);
    yend=acf(end);
elseif nargin < 4
    x=(point1:point2)*x_unit;
    y=acf(point1:point2);
    y0=acf(point1)-acf(point2);
    yend=acf(point2);
elseif nargin<5
    point2=length(acf);
    point1=2;
    %x=x(point1:point2);
    y=acf;
    y0=acf(1)-acf(end);
    yend=acf(end);
elseif nargin<6
    point2=length(acf);
    y=acf(point1:point2);
    y0=acf(point1)-acf(point2);
    yend=acf(point2);
end
if size(x,1)<size(x,2)
    x=x';
end
% start_t = [toff1,ton1,toff2,ton2]
% start_v = initial [a,b,c,d,e] for curve fitting
%
start_v=[y(end)*start_t(1)/start_t(2),1/(1/start_t(1)+1/start_t(2)),...
%
y(end)*start_t(3)/start_t(4),1/(1/start_t(3)+1/start_t(4)),y(end)];
start_v=start_t;
```

```
%%
dec2=fittype('(1+f1/(1-f1)*exp(-x/t1))*(1+f2/(1-f2)*exp(-
x/t2))','dependent','y');
fo=fitoptions(dec2);
fo.StartPoint=start_v;
fo.Lower=zeros(1,4);
fo.TolFun=1e-7;
fo.TolX=1e-7;
fo.MaxIter=1000;
[f,gof] = fit( x, y, dec2, fo);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% t1 = shorter time constant %%%
%%% t2 = longer time constant  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize a+b1+b2=1
% if f.t1<f.t2
%     b1=f.b1/(f.a+f.b1+f.b2);
%     t1=f.t1;
%     b2=f.b2/(f.a+f.b1+f.b2);
%     t2=f.t2;
%     a=f.a/(f.a+f.b1+f.b2);
% else
%     b2=f.b1/(f.a+f.b1+f.b2);
%     t2=f.t1;
%     b1=f.b2/(f.a+f.b1+f.b2);
%     t1=f.t2;
%     a=f.a/(f.a+f.b1+f.b2);
% end
% a,b1,t1,b2,t2
% t1<t2, k1>k2
e1=[num2str(1/f.t1),'=1/ton1+1/toff1'];
e2=[num2str(1/f.t2),'=1/ton2+1/toff2'];
e3=[num2str(f.f1/(1-f.f1)),'=toff1/ton1'];
e4=[num2str(f.f2/(1-f.f2)),'=toff2/ton2'];
s=solve(e1,e2,e3,e4,'toff1','toff2','ton1','ton2');
toff1=double(s.toff1);
toff2=double(s.toff2);
ton1=double(s.ton1);
ton2=double(s.ton2);
% Fisc1=double(s.kon1)/k_exc;Fisc1=Fisc1(2);
% Fisc2=double(s.kon2)/k_exc;Fisc2=Fisc2(2);
% p_param=[toff1,toff2;Fisc1,Fisc2];
t_acf=[toff1,ton1,toff2,ton2];
rsq=gof.adjrsquare;
```

```
function bic_value=BIC(logp,Ndata,numState)
% calculate BIC from likelihood, the number of data points, and number
of
% hidden states. The equation is explained in my thesis
bic_value=logp-(numState^2+2*numState-3)*log(Ndata)/2;
% experimental : add degree of freedom about intensity and internal
```

```
function ccf =
crosscorr_photon(seq1,seq2,lag_min,lag_max,lag_unit,lag_step)
% calculate cross correlation of two different data
% method = 0 : input is photon arrival time
% method = 1 : input is time-binned intensity trajectory
% method = 2 : input is the sparse array including the position of non-
zero
% photons
% lag_unit : This value makes photon arrival time to be sparse array,
If
% the input is photon arrival time, it is divided by the lag_unit, and
% treated as the position of photons.
%chan1=min(raw_data(:,5));
%chan2=max(raw_data(:,5));
%valid_photons = raw_data(raw_data(:,6)==0,3);
%seq1=raw_data(raw_data(:,5)==chan1,3);
%seq2=raw_data(raw_data(:,5)==chan2,3);
ccf=zeros(round(lag_max/lag_unit)-round(lag_min/lag_unit)+1,2);
ccf(:,1)=(lag_min:lag_unit:lag_max)';
if nargin <6
    lag_step=1;
end
% If round(seq1) is same as seq1, seq1 is photon arrival
% time, so method index will be 0. If seq1 has any zero, it is a
% time-binned photon counts, so the method index will be 1. Otherwise
the
% seq1 is a sparse array including photon positions, then the method
% index will be 2.

seq_ori1 = ceil(seq1/lag_unit);
seq_ori2 = ceil(seq2/lag_unit);
ccf(1,2)=length(intersect(seq_ori1,seq_ori2));
for cc=ceil(lag_min/lag_unit):ceil(lag_max/lag_unit)
    % shift photons positions by cc-1
    temp_counts_seq_delay = seq_ori2 + cc*lag_step;
    %temp_counts_seq_delay = seq_ori + cc-1;
    shift_to_head = temp_counts_seq_delay(temp_counts_seq_delay >
seq_ori2(end));
    counts_seq_delay = zeros(length(seq_ori2),1);
    counts_seq_delay(1:length(shift_to_head)) = shift_to_head-
seq_ori2(end);
    counts_seq_delay(length(shift_to_head)+1:end) =
temp_counts_seq_delay...
        (1:end-length(shift_to_head));
    %ccf(cc)=sum(temp_seq.*counts_seq)*length(counts_seq)/...
    %    (sum(counts_seq)^2*(length(counts_seq)-cc));
    %ccf(cc)=sum(temp_seq.*counts_seq)/length(counts_seq);
    ccf(cc-
ceil(lag_min/lag_unit)+1,2)=length(intersect(seq_ori1,counts_seq_delay)
);
    %sum_delay(cc)=sum(temp_seq);
end
plot(ccf(:,1),ccf(:,2));grid on
xlabel('\Deltat(s)','fontsize',13)
ylabel('C(t)','fontsize',13)
```

```
function dwell_seq=dwell_time(seq,method,dw_type)
% convert data into dwell time sequence or vice versa
% method 0 (default) :convert data into dwell time sequence
% method 1 :convert dwell time sequence into data sequence
% dwell_seq = [label, dwell time]
if size(seq,1)<size(seq,2)
    seq=seq';
end
if nargin  == 1 || method == 0
    s1=[seq;seq(end)-1];
    s2=[seq(1);seq(1:end)];
    %change_p2=find([[seq;seq(end)-1]-[seq(1);seq(1:end)]]~=0)-1;
    change_p2=find(s1-s2~=0)-1;
    dwell_seq=change_p2-[0;change_p2(1:end-1)];
    dwell_seq(:,2)=seq(change_p2);
else
    cum_dwell = cumsum(seq(:,1));
    L = cum_dwell(end);
    %     [~,m]=memory;
    %     if L>m.PhysicalMemory.Available-5e8
    %         error('You will have a memory problem.')
    %     else
    if (nargin<3) || (strcmp(dw_type,'int')==1)
        dwell_seq = uint8(zeros(L,1));
    elseif strcmp(dw_type,'double')==1
        dwell_seq = zeros(L,1);
    end
    dwell_seq(1:cum_dwell(1)) = seq(1,2);
    for cc=2:length(cum_dwell)
        dwell_seq(cum_dwell(cc-1)+1:cum_dwell(cc)) = seq(cc,2);
    end
end
end
```

```
function fit_param=extract_p_param_num(pw,state_seq,fit_tr,fit_e,...
    input_param,index_pwt)
%%% This code extracts dark state quantum yield and lifetime from
trained
%%% transition and emission matrices or reconstructed state sequence.
% The code returns toff or Fdark as zero if those are photophysically
% irrelevant.

% input parameters
% index_pwt : 0(default,pwt from emission matrix), 1(mean pwt of state
i)
% index_pwt=0 (recommended). Viterbi algorithm has intrinsic weakness
for
% short photon waiting time. Short photons at state 2 may be
misassigned as
% state 1. It should be improved.
%%%   Soonkyo Jung 05/10/2015



%%%% correction history
% removed lines for using the most populated state in estimating Fdarks
% use preassigned t_ic
% Fdark(cps) is estimated by trp(cps+1)/trp(cps) and Fdark(cps-1)
% toff(cps) is estimatmed by pwt(cps+1) and Fdark(cps)
% Fic is estimated using the sum of Fdark.
% correct formula about Ffl and internal conversion
% error messages about negative lifetime or quantum yield are suppresed
% and was replaced by warning.
% extract photophysical parameters(quantum yield, lifetime) from
% transition and emission matrix
% correct params about missed dark states


index_toff = 0;
if nargin<7
    index_toff=0;
    if nargin<6
        index_pwt=0;
    end
end
%% Structure of input and output photophysical parameters

numStates=size(input_param,2)+1;
t_bin=input_param(11,1);
[~,~,t_ground]=absc(input_param);
t_fl=input_param(6,1);
t_ic=input_param(13,1);
Ffl=input_param(4,1);
Fdet=input_param(5,1);
Fcol=input_param(4,1)*input_param(5,1);
back_level=input_param(10,1);
Fdark_fit=zeros(1,numStates-1);
toff_fit=zeros(1,numStates-1);
x=(1:length(fit_e))';
fit_param=input_param;
```

```
%% pwt : real photon waiting time of each state
pwt=zeros(1,numStates);
for cc=1:numStates
    if index_pwt==1
        pwt(cc)=t_bin*mean(pw(state_seq==cc));
    else
        pwt(cc)=t_bin*sum(x.*fit_e(:,cc));
    end
end
% estimate pwt from the inverse of waiting times subtracted by
background
% Those should be positive.
pwt=1./(1./pwt-back_level);
% pwt=abs(pwt);
%% ps1 -> sum of all Fdarks

if Fcol~=1
    %   Two ways to extract sum of dark state quantum yield
    %   1. from photon waiting time of ps1
    % pwt(1)=(t_ground+Fic*t_ic+Ffl*(1-Fdet)*tFL)/(Fdet*Ffl+Fdark)+tFL
    s=solve([num2str(pwt(1)-t_fl,6),'=(',num2str(t_ground+Ffl*t_fl*(1-
Fdet),6),...
        '+(1-',num2str(Ffl),'-
Fdark)*',num2str(t_ic),')/(',num2str(Fdet*Ffl),...
        '+Fdark)+',num2str(t_fl)],'Fdark');
    sumFdark_fit1=double(s);
    % 2. from transiton probabilityes trp(1,1)
    % Fdark from trp(1,1,no background)
    e_tr=[num2str(fit_tr(1,1)*(1+back_level*pwt(1))-
back_level*pwt(1),6),'=',...
        num2str(Fcol,5),'/(',num2str(Fcol,5),'+Fdark)'];
    s=solve(e_tr);
    sumFdark_fit2=double(s);

    if (sumFdark_fit1>0)+(sumFdark_fit1<1-Ffl)==2 &&
(sumFdark_fit2>0)+(sumFdark_fit2<1-Ffl)~=2
        % If Fdark from pwt is correct, and Fdark from transiton matrix
is
        % wrong, choose sumFdark_fit1.
        % sumFdark_fit1 : Fdark from pwt
        sumFdark_fit=sumFdark_fit1;
    elseif (sumFdark_fit1>0)+(sumFdark_fit1<1-Ffl)~=2 &&
(sumFdark_fit2>0)+(sumFdark_fit2<1-Ffl)==2
        % If Fdark from transition matrix is correct, and Fdark from
pwt is
        % wrong, choose sumFdark_fit2
        % sumFdark_fit2 : Fdark from tr
        sumFdark_fit=sumFdark_fit2;
    elseif (sumFdark_fit1>0)+(sumFdark_fit1<1-Ffl)==2 &&
(sumFdark_fit2>0)+(sumFdark_fit2<1-Ffl)==2
        % If both Fdark are correct, choose Fdark from pwt. Mostly this
one
        % is better.
        sumFdark_fit=sumFdark_fit2;
```

```
    elseif (sumFdark_fit1>0)+(sumFdark_fit1<1-Ffl)~=2 &&
(sumFdark_fit2>0)+(sumFdark_fit2<1-Ffl)~=2
        % both Fdark are wrong.
        fit_param=input_param;
        fit_param(7:8,:)=0;
        return
    end
else
    sumFdark_fit=1-fit_tr(1,1);
end
% Fic : internal conversion quantum yield
Fic=1-sumFdark_fit-Ffl;
% % another way
% s=solve([num2str(Fcol,4),'*(1-Fdark)/(',num2str(Fcol,4),'*(1-
Fdark)+Fdark)=',num2str(fit_tr(1,1),5'),'+',...
%     num2str(back_level*(fit_tr(1,1)-
1),5),'*',num2str(t_ground+t_fl,5),'/(',num2str(Fcol,4),'*(1-
Fdark)+Fdark)']);
% s=double(s);
% sumFdark_fit=s(s>0);
%% trp with no background -> All Fdark
%%%% transition probabilities with no background
%%%% N states, (N-1) dark states
% trp(1)=Fcol/dfs;
% trp(i)=Fdark(i-1)*Fcol/((dfs-sum(Fdark(1:i-2)))*(dfs-sum(Fdark(1:i-
1))))
% i=1,2,...,N

%%%% by background trp changes as follows
%%% new (good)
% trpnb(i,i)=1-(back_level*pwt(i)+1)*(1-trpb(i,i));
%%% old
% trpnb(i,i)=trpb(i,i)+back_level*pwt(i)*(trpb(i,i)-1);

%%% use most populated state to minimize error. Skipped now
% h=histo_HMM(state_seq);h(1)=[];
% h=fit_tr^1e5;h=h(1,:);
% max_state=argmax(h);
max_state=1;

% if index_pwt==1
%     pwti=t_bin*mean(pw(state_seq==max_state));
% else
%     pwti=t_bin*sum(x.*fit_e(:,max_state));
% end

trp=fit_tr(max_state,:);
% estimate P(singlet|singlet) with no background
trp_max=1-(back_level*pwt(max_state)+1)*(1-trp(max_state));

dfs=Fdet*Ffl+sumFdark_fit;

% use ps1 first
% sumFdark_fit
if max_state==1;
    % if numStates<3, it means only one dark state exists.
```

```
    if numStates>2
        % eqtrp : equation for Fdarks from trp(i)/trp(i+1).
i=2,...numStates-1
        eqtrp=[];
        % solve Fdark1 first
        s=solve([num2str(trp(2)/trp(1)),'=Fdark1/(',num2str(dfs),'-
Fdark1)'],'Fdark1');
        Fdark_fit(1)=double(s);
        if (Fdark_fit(1)>=1) || (Fdark_fit(1)<=0)
            %             Fdark_fit(1)=0;
            fit_param=input_param;
            fit_param(7:8,:)=0;
            return
        end
        for cps=2:numStates-1
            % simultaneous equations to solve trp for Fdarks

            % sumFdarki2 = sum(Fdark1~Fdark(i-2))
            sumFdarki2='(0';
            for cc=1:cps-2
                sumFdarki2=[sumFdarki2,'+Fdark',num2str(cc)];
            end
            sumFdarki2=[sumFdarki2,')'];
            % sumFdarki1 = sum(Fdark1~Fdark(i-1))
            sumFdarki1='(0';
            for cc=1:cps-1
                sumFdarki1=[sumFdarki1,'+Fdark',num2str(cc)];
            end
            sumFdarki1=[sumFdarki1,')'];
            % sumFdarki = sum(Fdark1~Fdarki)
            %             if cps>=3
            sumFdarki='(0';
            for cc=1:cps
                sumFdarki=[sumFdarki,'+Fdark',num2str(cc)];
            end
            sumFdarki=[sumFdarki,')'];
            % Instead of solving individual transition probabilities,
use
            % the ratio of trps of adjacent states.
            % trp(i)/trp(i+1). i=2,...,numStates-1
            % trp(i)/trp(i+1)=Fdark(i-1)*(dfs-sum(Fdark(1:i-1))-
Fdark(i))
            %  /Fdark(i)/(dfs-sum(Fdark(1:i-2)))
            % solve Fdark(cps) using trp(cps+1)/trp(cps) and Fdark(cps-
1)

eqcps=[num2str(trp(cps+1)/trp(cps)),'=Fdark',num2str(cps),...
                '*',num2str(dfs-sum(Fdark_fit(1:cps-
2))),'/',num2str(Fdark_fit(cps-1)),...
                '/(',num2str(dfs-sum(Fdark_fit(1:cps-1))),'-
Fdark',num2str(cps),')'];
            s=solve(eqcps,['Fdark',num2str(cps)]);
            if length(s)==1
                Fdark_fit(cps)=double(s);
            else
                Fdark_temp=double(s);
```

```
Fdark_fit(cps)=Fdark_temp(((Fdark_temp<1)+(Fdark_temp>0))==2);
            end
        end
    else
        Fdark_fit=sumFdark_fit;
        if (Fdark_fit>=1) || (Fdark_fit<=0)
            Fdark_fit=0;
        end
    end
else

end

%% trp : state weight will be described in trp, not Fdark.
ds=Fcol*(1-sum(Fdark_fit));
dss=Fcol*(1-sum(Fdark_fit))+sum(Fdark_fit);
trp=[1-dss,ds,Fdark_fit];
% solve toff using mean photon waiting times
if index_toff==0
    %% ps(cps) -> toff(cps-1), i=3,4,...numStates
    % single equations for toff(1~numState-1)
    %     qy=[1,Fic,Ffl*(1-Fdet),Fdark,Ffl*Fdet];
    % lifetime=[t_ground,t_ic,t_fl,toff,t_fl];
    % cps=2;
    % p1=sum(qy(2:cps+1));
    % w1=[1-p1;p1];
    % t1=[t_ground+toff(cps-1);...
    %     sum(qy(2:3).*(lifetime(2:3)+t_ground))/p1/(1-
p1)+t_ground+toff(cps-1)];
    % p2=sum(qy(2:cps+1))+Fdark(cps-1);
    % w2=[1-p2;p2];
    %
t2=[t_ground+t_fl;(sum(qy(2:3).*(lifetime(2:3)+t_ground))+toff(cps-
1)*Fdark(cps-1))/p2/(1-p2)+t_ground+t_fl];


    for cps=2:numStates
        Fsinglet=Fic+Ffl*(1-Fdet);
        p1=Fsinglet+sum(Fdark_fit(1:cps-2));
        w11=['(',num2str(1-p1),')'];
        w12=['(',num2str(p1),')'];
        t11=['(',num2str(t_ground),'+toff',num2str(cps-1),')'];
        t12=['(',num2str(sum([Fic,Ffl*(1-
Fdet)].*([t_ic,t_fl]+t_ground))/p1/(1-p1)+t_ground),...
            '+toff',num2str(cps-1),')'];
        p2=Fsinglet+sum(Fdark_fit(1:cps-1));
        w21=['(',num2str(1-p2),')'];
        w22=['(',num2str(p2),')'];
        t21=['(',num2str(t_ground+t_fl),')'];
        t22=['(',num2str(sum([Fic,Ffl*(1-
Fdet)].*([t_ic,t_fl]+t_ground))/p2/(1-p2)),'+',...
            num2str(t_ground+t_fl),'+toff',num2str(cps-1),'*',...
            num2str(Fdark_fit(cps-1)/p2/(1-p2)),')'];
```

```
eq_e=[w11,'*',w21,'*(',t11,'+',t21,')+',w12,'*',w21,'*(',t12,'+',t21,')
+',...

w11,'*',w22,'*(',t11,'+',t22,')+',w12,'*',w22,'*(',t12,'+',t22,')==',nu
m2str(pwt(cps))];
        toff_var=['toff',num2str(cps-1)];
        s=solve(eq_e,toff_var);
        % if the solution is infinity, zero, or does not exist, return
zero
        s_temp=double(s);
        if isempty(s_temp)
            % if the solution for toff_cps does not exist, return zero.
            toff_fit(cps-1)=0;
        elseif length(s_temp)>1
            % if the multiple solutions for toff_cps exist, return
positive
            % one.
            toff_fit(cps-1)=s_temp(s_temp>0);
        elseif (s_temp==0) || (s_temp==inf) || (s_temp<0)
            % if the solution for toff_cps is zero, infinite, or
negative,
            % return zero.
            toff_fit(cps-1)=0;
        else
            % if the solution for toff_cps is a positive number, return
it.
            toff_fit(cps-1)=s_temp;
        end
    end

elseif index_toff>0
    % dwell time at ps instead of individual photon waiting times
    dw=dwell_time_state(cumsum(pw),state_seq);
    pdwt=zeros(numStates,1);
    for cps=1:numStates
        pdwt(cps)=mean(dw(dw(:,2)==cps,1));
    end
    clear dw
    %% ps(i) -> toff(i-1)
    for cps=2:numStates
        eq_pdwt=[num2str(t_bin*pdwt(cps)),'=((2*(',num2str(dss),'-
',num2str(sum(Fdark_fit(1:cps-1)))...
            ,')+',num2str(Fdark_fit(cps-1)),')*',num2str((t_ground+(1-
dss)*t_fl+sum(Fdark_fit(1:cps-2).*toff_fit(1:cps-2)))),...
            '+(',num2str(dss),'-',num2str(sum(Fdark_fit(1:cps-
2))),')^2*toff',num2str(cps-1),')/',...
            num2str(2*ds*sum(Fdark_fit(cps:numStates-
1))+sum(Fdark_fit(cps:numStates-1))*sum(Fdark_fit(cps-1:numStates-
1))+ds^2)];
        s=solve(eq_pdwt,['toff',num2str(cps-1)]);
        %     toff_fit(cps-1)=eval(['double(toff',num2str(cps-1),')']);
        toff_fit(cps-1)=double(s);
        if double(s)<0
            warning('Negative lifetime')
        end
    end
```

```
end

%% check p_param again
% Dark state quantum yields should be less than one. Ffl+Fdarks should
be
% positive less than one. toffs should be positive.
if sum(Fdark_fit)+fit_param(4,1) > 1
    Fdark_fit(:)=0;
end
if sum(toff_fit<0)~=0
    toff_fit(:)=0;
end
%% return fit_param
fit_param=input_param;
fit_param(7,:)=toff_fit;
fit_param(8,:)=Fdark_fit;



function text_sumFdark=eq_sumFdark(n)
% Fdet=input_param(5,1)*input_param(4,1);
% Fdark=sum(input_param(8,:));
% dss=Fdet*(1-Fdark)+Fdark;
% if n==-1
%     text_sumFdark=num2str(dss-1);
% elseif n >= 0
text_sumFdark='(0';
for cc=1:n
    text_sumFdark=[text_sumFdark,'+Fdark',num2str(cc)];
end
text_sumFdark=[text_sumFdark,')'];
% end
```

```
function p=exppdf_back(x,t_back,decay_t)
% Probability distribution of the convolution of exponential
distributions
% superposed with poisson noise.
% analytical formula. based on superposition of renewal processes
% decay_t : time constants of convoluted exponential distributions
% t_back : waiting time of poisson noise.
L=length(decay_t);
p=zeros(size(x));
if t_back==inf
    decay_t=decay_t(decay_t>0);
    decay_t=decay_t(isinf(decay_t)==0);
    p=exppdf_conv(x,decay_t);
else
    for cc=1:L

        %%% change decay time by small amount to avoid NaN error
        dw=dwell_time(sort(decay_t));
        if sum(dw(:,1)~=1)~=0
            % convolution of Erlang and exponential distribution is not
done yet.
            % now change repetitive time constants by very small random
amount.
            t_new=dw(dw(:,1)==1,2);
            lr=find(dw(:,1)~=1);
            for ct=1:length(lr)
                t_new=[t_new;dw(lr(ct),2)*(1+(0:dw(lr(ct),1)-1)'*1e-
5)];
            end
            decay_t=t_new;
        end

        t_temp=decay_t;
        t_temp(cc)=[];
        p=p+exp(-x/decay_t(cc))*decay_t(cc)^(L-
2)*(decay_t(cc)+t_back)^2/...
            prod(decay_t(cc)-t_temp);
    end
    p=p.*exp(-x/t_back)/t_back/(sum(decay_t)+t_back);
end
```

```
function pdf_conv=exppdf_conv(x,t)
% pdf_conv : convolution of exponential distributions.
% t ; time constants of convoluted exponential distributions
% all t should be different from each other.
t=sort(t);
% make t as a column vector.
if size(t,1)<size(t,2)
    t=t';
end
if size(x,1)<size(x,2)
    x=x';
end
t(t==0)=[];
t(isinf(t)==1)=[];
dw=dwell_time(sort(t));
if sum(dw(:,1)~=1)~=0
    % convolution of Erlang and exponential distribution is not done
yet.
    % now change repetitive time constants by very small random amount.
    t_new=dw(dw(:,1)==1,2);
    lr=find(dw(:,1)~=1);
    for cc=1:length(lr)
        t_new=[t_new;dw(lr(cc),2)*(1+(0:dw(lr(cc),1)-1)'*1e-5)];
    end
    t=t_new;
end


if isempty(t)==0
    % dw=dwell_time(t);
    % n_conv : number of exponentials for convolution
    n_conv=length(t);
    pdf_conv=zeros(length(x),1);
    pdf_zero=0;
    if n_conv<2
        pdf_conv=exppdf(x,t);
    else
        for cc=1:n_conv
            t_temp=t;
            t_temp(cc)=[];
            pdf_conv=pdf_conv+exp(-x/t(cc))*t(cc)^(n_conv-2)/...
                prod(repmat(t(cc),n_conv-1,1)-t_temp);
            pdf_zero=pdf_zero+t(cc)^(n_conv-2)/...
                prod(repmat(t(cc),n_conv-1,1)-t_temp);
        end
    end
    % correct negative probability. Due to unknown reason(maybe small
number
    % error), first few probability values are negative. Shift them to
    % positive.
    pdf_conv=pdf_conv-pdf_zero;
    if sum(pdf_conv<0)~=0
        pdf_conv=pdf_conv-min(pdf_conv);
    end
    % remove NaN
    pdf_conv(isnan(pdf_conv))=0;
```

```
else
    pdf_conv=zeros(size(x));
end
```

```
function ov_index = ov_ranksum(seq, state_seq)
% check overlap with Wilcoson rank sum test
% seq : samples
% state_seq : states of samples
% ov_index ; label of overlapping states
% work well for exponential distribution
% work a little worse for normal distribution (relative efficiency :
0.955)
if min(state_seq) == 0
    error('State Zero exists!!!')
    %state_seq=state_seq+1;
end
L=max(state_seq);
% L : number of state
ov_index_temp = zeros(1,L*(L+1)/2);
cell_seq=cell(L,1);
for cc=1:L
    cell_seq{cc}=seq(state_seq==cc);
end
count_s1=1;
cc_temp = 1;
while count_s1<L
    for count=count_s1+1:L
        if isempty(cell_seq{count})==0
            % if state(count) exist
            [~,h] = ranksum(cell_seq{count_s1},cell_seq{count});
            if h == 0
                ov_index_temp(cc_temp) = count;
                cc_temp = cc_temp+1;
                list_temp = ov_index_temp(ov_index_temp~=0);
                count_s1 = list_temp(end);
            else
                %ov_index_temp(cc_temp) = count;
                %cc_temp = cc_temp+1;
                count_s1 = count;
                break;
            end
            % if state(count) is empty
        else
        end
    end

end
ov_index = ov_index_temp(ov_index_temp~=0);
```

```
function [tr,e,pwt_back,pops,intensity]=extract_tre(p_param,numEmis)

% Estimate transition and emission matrix and photon waiting times from
% photophysical parameters(p_param).
% numEmis : the size of emission matrix
% pwt_back : expected photon waiting time of each state.
% pops : mean accessed number of each photophysical processes during
photon
% waiting time of given ps
% intensity : expected photon count (/s)

% correction :Ffl and internal conversion were introduced correctly.
% Need for separate photon missing and internal conversion
numStates=size(p_param,2)+1;
pops=zeros(numStates,numStates+3);
Fdark=p_param(8,:);
Ffl=p_param(4,1);
Fdet=p_param(5,1);
% Fic : the quantum yield of internal conversion
Fic=1-sum(Fdark)-Ffl;
Fcol=Ffl*Fdet;
back_level=p_param(10,1);
[~,~,t_ground]=absc(p_param);
t_fl=p_param(6,1);
t_ic=p_param(13,1);
toff=p_param(7,:);
t_bin=p_param(11,1);
dfs=Ffl*Fdet+sum(Fdark);
if nargin<2
    numEmis=10;
end
%% extract tr with no background
% tr=zeros(numStates);
tr_nb=zeros(numStates);
tr_nb(:,1)=Ffl*Fdet/dfs;
tr_nb(:,2)=Ffl*Fdet*Fdark(1)/(dfs*(dfs-Fdark(1)));
% tr_nb(:,numStates)=Fdark(numStates-1)/(Fdark(numStates-1)+ds);
for cps=2:numStates
    tr_nb(:,cps)=Ffl*Fdet*Fdark(cps-1)/((dfs-sum(Fdark(1:cps-2)))*...
        (dfs-sum(Fdark(1:cps-1))));
end
% introduce background
%% photon waiting time of hidden states without noise
% quantum yield(qy)=[ground, internal conversion, photon missing, dark
% states, photon detection]
qy=[1,Fic,Ffl*(1-Fdet),Fdark,Ffl*Fdet];
% lifetime = [ground, t_IC, t_Fl, toffs, t_Fl]
lifetime=[t_ground t_ic t_fl p_param(7,:)  t_fl];
pwt=zeros(numStates,1);
p=sum(qy(2:3));
pwt(1)=sum(qy(1:3).*lifetime(1:3))/(1-p)+t_fl;
% pops : population photophysical processes before one photon is
detected.
pops(:,end)=1;
pops(1,1:3)=qy(1:3)/(1-p);
for cps=2:numStates
```

```
    % pwt(cps)=dwell_time(pre)+dwell_time(post)
    % dwell_time(pre) = P(zero waiting,pre)*(ground+toff(cps-1))+
    %       P(nonzero waiting) * waiting time(one period of dark(cps-
1))
    % dwell_time(post) = P(zero waiting,post)*(ground+t_fl)+
    %       P(nonzero waiting)*waiting time(including dark(cps-1))

    p1=sum(qy(2:cps+1));
    p2=sum(qy(2:cps+2));
    w1=[1-p1;p1];
    if dfs~=1
        t1=[t_ground+toff(cps-
1);sum(qy(2:cps+1).*(lifetime(2:cps+1)+t_ground))...
            /p1/(1-p1)+t_ground+toff(cps-1)];
    else
        t1=[t_ground+toff(cps-1);0];
    end
    w2=[1-p2;p2];
    t2=[t_ground+t_fl;sum(qy(2:cps+2).*(lifetime(2:cps+2)+t_ground))...
        /p2/(1-p2)+t_ground+t_fl];
    pwt(cps)=sum((repmat(w1,2,1).*reshape(repmat(w2',2,1),4,1)).*...
        (repmat(t1,2,1)+reshape(repmat(t2',2,1),4,1)));
    pops(cps,1:cps+2)=[sum(qy(2:cps+1)/(1-p1))+1,qy(2:cps+1)/(1-
p1),1]...
        +[sum(qy(2:cps+2)/(1-p2)),qy(2:cps+2)/(1-p2)];
end
%% modify tr for background noise
if back_level>0
    % fit_tr(i,i)+(back_level*fit_tr(i,i)-back_level)*pwt
    tr=tr_nb;
    for cps=1:numStates
        % staying probability is increased by the number of background
        % photons.
        tr(cps,cps)=1-(1-tr_nb(cps,cps))/(back_level*pwt(cps)+1);
        trp_ratio=tr_nb(cps,:);
        trp_ratio(cps)=0;
        % escaping probability is 1-tr(cps,cps). After escape from the
same
        % state, weight of next state is constant with or without
        % background noise.
        trp_ratio=trp_ratio/sum(trp_ratio)*(1-tr(cps,cps));
        trp_ratio(cps)=tr(cps,cps);
        tr(cps,:)=trp_ratio;
    end
    tr=norm_m(tr);
    pwt_back=1./(1./pwt+back_level);
else
    pwt_back=pwt;
    tr=norm_m(tr_nb);
end
%% extract e
e=zeros(numEmis,numStates);
x=(1:numEmis)';

%%% state 1
% population of photophysical transitions until photon detection
```

```
% follows geometric distribution. Geometric random number has nonzero
% probability at zero, so transition prior to photon detection can be
% zero. Resulting dwell time is the convolution of waiting time before
% photon transition, one period of ground state, and singlet decay with
% a detected photon
% e(:,1)=dfs*exppdf_conv(x,[t_ground t_fl]/t_bin)+...
%     (1-dfs)*exppdf_back(x,1/back_level/t_bin,[(1/dfs-1)*t_ground,
t_fl*Ffl*(1-Fdet)/dfs]/t_bin);
% p for geopdf(ic)=1/(mean(ic)+1)=1/(Fic/(Fic+Ffl*(1-Fdet)*(1/(1-p)-1)
% =1/(Fic/(1-p)-1)
p=Fic+Ffl*(1-Fdet);
% Following formula is correct. However, due to unknown reason, the
% emission probabilities are stuck to some specific value. It should be
% corrected later. Instead of the correct formula, an approximate one
with
% exppdf of the sum of mean value is now used.
% e(:,1)=(1-p)*exppdf_conv(x,[t_ground,t_fl]/t_bin)+...
%     p*exppdf_conv(x,[(t_ic+t_ground)*Fic/p/(1-p)+...
%     (t_fl+t_ground)*Ffl*(1-Fdet)/p/(1-p),t_ground,t_fl]/t_bin);
e(:,1)=exppdf(x,((1-p)*(t_ground+t_fl)+p*((t_ic+t_ground)*Fic/p/(1-
p)+...
    (t_fl+t_ground)*Ffl*(1-Fdet)/p/(1-p)+t_ground+t_fl))/t_bin);
e(:,1)=e(:,1)/sum(e(:,1));

%%% state 2
% population(ground state) >= 2 (one for photon detection, another for
dark
% state transition)
% now exact convolution will be considered. (not weighted mean)
% t = [shape, scale]
% pdf for dwell time at each photophysical state is gamma distribution.
% weight : population of distribution
% timec : time constant of distribution
% 1/p1 and 1/p2 : mean number of transition to escape to darkEnd and
photon
% detection respectively.
% cps : count for photon state = 2
% singlet
if dfs~=1
    % p1: waiting probability before = Fic+Ffl*(1-Fdet)
    p1=sum(qy(2:3));
    % pdf of pre-dwell time is convolution of ground+dark1 and dwell
time
    % at Internal conversion,photon missing.

    w1=[1-p1,p1];

t1=[{[t_ground,toff(1)]},{[sum(qy(2:3).*(lifetime(2:3)+t_ground))/p1/(1
-p1)...
        ,t_ground,toff(1)]}];
    % pdf of pre-dwell time is convolution of ground+dark1 and dwell
time
    % at Internal conversion,photon missing.
    % p2: waiting probability (post) = Fic+Ffl*(1-Fdet)+Fdark1
    p2=sum(qy(2:4));
    w2=[1-p2,p2];
```

```
t2=[{[t_ground,t_fl]},{[sum(qy(2:4).*(lifetime(2:4)+t_ground))/p2/(1-
p2),...
        t_ground,t_fl]}];
    [~,weights_new,times_new]=multiexp_conv(x,w1,w2,t1,t2);

    for cc=1:length(weights_new)

e(:,2)=e(:,2)+weights_new(cc)*exppdf_back(x,1/back_level/t_bin,...
            cell2mat(times_new(cc,:))/t_bin);
    end
    e(:,2)=e(:,2)/sum(e(:,2));
else
    %     not done yet
    weight_singlet=[0;0;1];
    timec_singlet=[{0};{0};{t_fl}];
end

%%% state k (k>2)
if numStates>2
    for cps=3:numStates
        % p1 : P(waiting) = IC,PM,dark1
        p1=Fic+Ffl*(1-Fdet)+sum(Fdark(1:cps-2));
        w1=[1-p1,p1];
        t1=[{[t_ground,toff(cps-1)]},...
            {[sum(qy(2:cps+1).*(lifetime(2:cps+1)+t_ground))/p1/(1-
p1),t_ground,toff(cps-1)]}];
        % p2 : P(waiting) = IC,PM,dark1,dark2,photon detection
        p2=Fic+Ffl*(1-Fdet)+sum(Fdark(1:cps-1));
        w2=[1-p2,p2];
        t2=[{[t_ground,t_fl]},...
            {[sum(qy(2:cps+2).*(lifetime(2:cps+2)+t_ground))/p2/(1-
p2),t_ground,t_fl]}];
        [~,weights_new,times_new]=multiexp_conv(x,w1,w2,t1,t2);
        for cc=1:length(weights_new)

e(:,cps)=e(:,cps)+weights_new(cc)*exppdf_back(x,1/back_level/t_bin,...
                cell2mat(times_new(cc,:))/t_bin);
        end
        %         e(:,cps)=geoexp_conv(x,p_conv,timec_conv);
        e(:,cps)=e(:,cps)/sum(e(:,cps));
    end
end
%% intensity : an expected photon count
trp=tr^1e5;
trp=trp(1,:);
intensity = 1/sum(trp'.*pwt_back);
```

```
function [tr_multi, e_multi,
fit_time_multi,pi_multi,logp]=fit_multi_sequence(cell_seq,...
    cell_tr, cell_e, cell_fit_time,cell_pi,input_param,select_weights)
% Re-estimate transition and emission matrix from  multiple
observations
% fit_time_multi : mean photon waiting time / bin_time
% cell_seq : multiple dataset
% cell_tr, cell_e, cell_fit_time : trained tr,e, and time from each
dataset
% at given NumState
% cell_tr : trained e from each dataset at given NumState
% select_weights : input 'likelihood' or 'unit'
% weights can be either likelihood or unit(1).
%% photophysical parameters
Fdet=input_param(5,1);
toff=input_param(7,:);
Fdark=input_param(8,:);
Fdark_sum=sum(input_param(8,:));
I_prim=input_param(1,1);
s_abs=input_param(2,1)*3.82356e-21;
wavelength=input_param(3,1);
t_fl=input_param(6,1);
t_ic=input_param(13,1);
t_bin = input_param(11,1);
back_level=input_param(10,1);
I_sat = 6.6261e-34*2.9979e8/(s_abs*wavelength*10^-9*t_fl);
%Nbins = t_mod/tbin;
%% dimension of tr
dim_param=[length(cell_seq),length(cell_tr),length(cell_fit_time),lengt
h(cell_e)];
sum_dim=0;
for c_data=1:4
    sum_dim=sum_dim+sum(abs(dim_param-dim_param(c_data)));
end
% check the number of states of each fitting results
L_data=length(cell_seq);
mean_st=zeros(L_data,1);
for cc=1:L_data
    mean_st(cc)=length(cell_tr{cc});
end
mean_st=round(mean(mean_st(mean_st>0)));

if sum_dim~=0
    error('The number of observation does not match to one of inputs.')
else
    for cc=1:L_data
        numState(cc,1)=length(cell_tr{cc});
    end
    if sum(numState)~=0
        % if any fitting result exist from individual fitting,
continue.
        numState=mean(numState(numState~=0));
        tr_multi_dom=zeros(L_data,numState,numState);
        tr_multi_denom=zeros(L_data,numState,numState);
        logp=zeros(L_data,1);
        numE=zeros(L_data,1);
        for c_data=1:L_data
```

```
            numE(c_data)=max(cell_seq{c_data});
        end
        %     e_multi_dom=zeros(L_data,max(numE),numState);
        % e_multi_denom=zeros(L_data,max(numE),numState);
        e_multi_dom=zeros(max(numE),numState);
        e_multi_denom=zeros(max(numE),numState);
        fit_time_multi=zeros(mean_st,1);
        weights=zeros(L_data,1);
        for c_data=1:L_data
            if isempty(cell_tr{c_data})==0 &&
length(cell_tr{c_data})==mean_st...
                    && isempty(cell_seq{c_data})==0
                % exclude bad fitting and fitting results with
different number
                % of state from most results
                c_data
                %%% reduce seq larger than the length of emission
matrix to
                %%% avoid an error
                seq=cell_seq{c_data};
                e=cell_e{c_data};
                seq(seq>length(e))=length(e);
                %%% make emission prob of nonexisting bin to zero
                h=histo_HMM(seq);h(1)=[];
                e(h==0,:)=0;
                e=e./repmat(sum(e,1),size(e,1),1);
                [~,logp(c_data),alpha,beta] =
hmmdecode_sb(seq,cell_tr{c_data},e,...

1./(1./sum(cell_fit_time{c_data},2)/t_bin+back_level)/t_bin,...
                    cell_pi{c_data});
                % tr_multi = tr_multi_dom./tr_multi_denom
                % tr
                for i=1:numState
                    for j=1:numState
                        % dominator of tr_multi =
                        %
sum(alpha(t,i)*tr(i,j)*e(pw,j)*beta(t+1,j))/likelihood;
                        tr_multi_dom(c_data,i,j)=sum(alpha(1:end-
1,i)*cell_tr{c_data}(i,j)...

.*cell_e{c_data}(seq(2:end),j).*beta(2:end,j));
                        % denominator of tr_multi =
                        % sum(alpha(1:end-1)*beta(1:end-1))
                        tr_multi_denom(c_data,i,j)=sum(alpha(1:end-
1,i).*beta(1:end-1,i));
                    end
                end
                % e : Rabiner's
                %             for state=1:numState
                %                 for cp=1:length(cell_seq{c_data})-1
                %
e_multi_dom(c_data,cell_seq{c_data}(cp),state)=...
                %
e_multi_dom(c_data,cell_seq{c_data}(cp),state)+...
                %
alpha(cp,state)*beta(cp,state);
```

```
                %                 end
                %                 e_multi_denom(c_data,:,state)=...
                %
sum(alpha(:,state).*beta(:,state));
                %             end

                % use gamma instead of alpha*beta
                gamma=alpha.*beta;
                gamma=gamma./repmat(sum(gamma,2),1,numState);

                if strcmp(select_weights,'likelihood')==1
                    % select weights : unit, logp
                    weights(c_data)=logp(c_data);
                elseif strcmp(select_weights,'unit')==1
                    weights(c_data)=1;
                end

                for state=1:numState
                    for cp=1:length(cell_seq{c_data})
                        e_multi_dom(seq(cp),state)=...
                            e_multi_dom(seq(cp),state)...
                            +weights(c_data)*gamma(cp,state);
                    end
                    %
e_multi_denom(c_data,:,state)=sum(gamma(:,state));
                    e_multi_denom(:,state)=e_multi_denom(:,state)+...
                        weights(c_data)*sum(gamma(:,state));
                end

                % average fit_time
                fit_time_multi=fit_time_multi+cell_fit_time{c_data};
            else
                % if tr or e does not exist due to bad fitting, skip
them
                weights(c_data)=0;
                tr_multi_dom(c_data,:,:)=0;
                tr_multi_denom(c_data,:,:)=0;
                %             e_multi_dom(c_data,:,:)=0;
                %             e_multi_denom(c_data,:,:)=0;
            end
        end
        % To avoid enormous number, divide weights by maximum one.
        weights=exp(weights-max(weights));
        % average tr
        p1=0;
        p2=0;
        for c_data=1:L_data
            p1=p1+weights(c_data)*tr_multi_dom(c_data,:,:);
            p2=p2+weights(c_data)*tr_multi_denom(c_data,:,:);
        end

        tr_multi=norm_m(reshape(p1./p2,numState,numState));
        % average e
        % p1 : dominator of averaged e
        % p2 : denominator of averaged e
        p1=0;
```

```
            p2=0;
            for c_data=1:L_data
                p1=p1+weights(c_data)*e_multi_dom(c_data,:,:);
                p2=p2+weights(c_data)*e_multi_denom(c_data,:,:);
            end
            e_multi=e_multi_dom./e_multi_denom;
            % average pwt
            x=(1:length(e_multi))';
            fit_time_multi=sum(repmat(x,1,numState).*e_multi)*t_bin;
            %     fit_time_multi=fit_time_multi/L_data;
            % average pi
            pi_multi=zeros(1,numState);
            for c_data=1:L_data
                if isempty(cell_pi{c_data})==0
                    pi_multi=pi_multi+cell_pi{c_data};
                end
            end
            pi_multi=pi_multi/sum(pi_multi);
        else
            % if all individual fitting failed, terminate.
            tr_multi=[];
            e_multi=[];
            fit_time_multi=[];
            pi_multi=[];
            logp=-inf;
        end
    end
```

```
function [pdfs,weight,timec,pops]=geoexp_conv(x,p,t,ground)
% convolution of pdfs of geometric-populated exponential random
numbers.
% p : prob of event, t : time constant of Exppdf
% p and t should be arranged in their order.
% weight = [zero1+zero2,
nonzero1+zero2,zero1+nonzero2,nonzero1+nonzero2]
% pops : population of exponential events.
% p(1) : probability that the pre-event occurs
% p(2) : probability that the post-event occurs
% n_conv : the number of geoexp distribution to be convolved
if nargin <4
    ground=0;
end
n_conv=length(p);
pdfs=zeros(size(x));
% l_conv : the length of weight and time constant of resulting pdfs
l_conv=2^n_conv;
if n_conv==1
    pdfs=(1-p)*exppdf_conv(x,[t(1),ground]/p); % nonzero
    pdfs(1)=p/x(1);
    weight=[p;1-p];
    timec=[{0};{[t(1),ground]/p(1)}];
    pops=[{[0,0]};{[1/p(1),0]}];
elseif n_conv==2
    pdfs=p(2)*(1-p(1))*exppdf_conv(x,[t(1),ground]/p(1))+... % nonzero
t1
        p(1)*(1-p(2))*exppdf_conv(x,[t(2),ground]/p(2))+... % nonzero
t2
        (1-p(2))*(1-
p(1))*exppdf_conv(x,[[t(1),ground]/p(1),[t(2),ground]/p(2)]); % both
nonzero
    pdfs(1)=prod(p)/x(1);
    weight=[p(1)*p(2);p(2)*(1-p(1));p(1)*(1-p(2));(1-p(2))*(1-p(1))];
    timec=[{0};{[t(1),ground]/p(1)};{[t(2),ground]/p(2)};...
        {[[t(1),ground]/p(1),[t(2),ground]/p(2)]}];
    pops=[{[0,0]};{[1/p(1),0]};{[0,1/p(2)]};{[1/p(1),1/p(2)]}];
else
    weight=[p(1);1-p(1)];
    timec=[{0};{[t(1),ground]/p(1)}];
    pops=[1/p(1)-1;0];
    for cc=2:n_conv
        if p(cc)~=1
            weight=repmat(weight,2,1).*...
                reshape(repmat([p(cc),1-
p(cc)],size(weight,1),1),size(weight,1)*2,1);
            timec=[repmat(timec,2,1),...

reshape(repmat([{0},{[t(cc),ground]/p(cc)}],size(timec,1),1),size(timec
,1)*2,1)];
            if cc~= (fix(cc/2)*2)
                % c is odd : pre transitions
                pops=pops+[1/p(cc)-1;0];
            else
                % c is even : post transitions
                pops=pops+[0;1/p(cc)-1];
            end
```

```
        else
            % zero p : no waiting.
            weight=repmat(weight,2,1).*...

reshape(repmat([0.5,0.5],size(weight,1),1),size(weight,1)*2,1);
            timec=[repmat(timec,2,1),...

reshape(repmat([{[t(cc),ground]},{[t(cc),ground]}],size(timec,1),1),siz
e(timec,1)*2,1)];
            if cc~= (fix(cc/2)*2)
                % c is odd : pre transitions
                pops=pops+[1;0];
            else
                % c is even : post transitions
                pops=pops+[0;1];
            end
        end
    end
    for cc=1:l_conv
        if sum(cell2mat(timec(cc,:)))>0
            pdfs=pdfs+weight(cc)*exppdf_conv(x,cell2mat(timec(cc,:)));
        end
    end
end
if length(weight)~=l_conv
    error('Somethings wrong.')
end
```

```
function f=geopdf_conv(x,p,index)
% index ; determines whether random numbers include zero or not
% no index or [0,0] : x=0,1,2,...
% [1 0] or [0 1] : x1= 0,1,2,... and x2=1,2,3,...
% [1 1] : x= 2,3,4,...
if nargin<3 || sum(index)==0
    x(x<0)=[];
    if length(p)==2
    f=p(1)*p(2)/(p(1)-p(2))*(1-p(2)).^(x+1).*(1-((1-p(1))/(1-
p(2))).^(x+1));
    elseif length(p)==3
        f=p(1)*p(2)*p(3)*((1-p(1)).^(x+1)/(p(1)-p(2))/(p(1)-p(3)))...
            +(1-p(2)).^(x+1)/(p(2)-p(1))/(p(2)-p(3)))...
            +(1-p(3)).^(x+1)/(p(3)-p(1))/(p(3)-p(2)));
    end
elseif sum(index)==1
    x(x<1)=[];
    f=p(1)*p(2)/(p(1)-p(2))*((1-p(2)).^x-(1-p(1)).^x);
elseif sum(index)==2
    x(x<2)=[];
    f=p(1)*p(2)/(p(1)-p(2))*(1-p(2)).^(x-1).*(1-((1-p(1))/(1-
p(2))).^(x-1));
end
% p1=p(1);
% p2=p(2);
% f=p1*p2/(p1-p2)*(1-p2).^(x+1).*(1-((1-p1)/(1-p2)).^(x+1));
```

```
function [histogram, x_hist] = histo_HMM(seq, bin)
%%% calculate histogram of discrete data
%%% bin = 1
%%% length of result = max of seq + 1
%%% x_hist : x axis of histogram ( 0 <= x_hist <= max(seq)
numStep1 = size(seq,1);
numStep2 = size(seq,2);
if nargin < 2
    bin = max(max(seq));
end
x_hist = 0:max(max(seq));
histogram = zeros(bin+1,1);
for count1 = 1:numStep1
    for count2 = 1:numStep2
%         if seq(count1,count2) == 0
%             histogram(1) = histogram(1) + 1;
%         else
            histogram(seq(count1,count2)+1) =
histogram(seq(count1,count2)+1) + 1;
%         end
    end
end
```

```
function [photon, ps,ps2]= hmmphotons(input_param,sim_mode)
% Generate photon trajectory from photophysical parameters by
Gillespie's
% algorithm
% photon : photon arrivall time
% ps : state of photons
% correction : mean dwell time at ground state and excitation rate
% fluorescence quantum yield is considered.
% time scale of internal conversion is ~fs or ~ps. It's ignorable in ns
or
% microsecond scale.
% rate parameter   state
%    t_on1,t_on2,... : on time
%    t_off1,t_off2,... : on time
%    k_off : escape rate from off to on

% state = 1 : ground
% state = 2 : singlet(internal conversion)
% state = 3 : singlet(photon missing)
% state = 4 : singlet(photon detection)
% state = 5 : dark states

% input parameters should be in following orders.
% 1. I_prim : Primary excitation intensity (W/cm2)
% 2. e : absorption coefficient (/(M*cm))
% 3. wavelength (nm)
% 4. Ffl : fluorescence quantum yield
% 5. Fdet : detection efficiency
% 6. t_fl : "measured" fluorescence lifetimea
% 7. dark state lifetime : t_dark1(shorter,T1_1 -> S0),
t_dark2(longer,T1_2 -> S0)
% 8. dark state quantum yield : Shorter- : k_ISC1(intersystem crossing
rate)/k_rad
% Longer-dark state quantum yield : k_ISC2(intersystem crossing
rate)/k_rad
% 9. ratio of k_ReISC (reverse intersystem crossing rate) to k_ISC2
%    (triplet depopulation rate from T1 to S0) : k_ReISC/k_ISC2
%    - Action cross section for reverse intersystem crossing due to
%     secondary laser illumination
% 10. background noise
% 11. t_bin : time step
% 12. collecting time(time mode) or collecting number of photons
(photon
% mode)
% 13. t_ic : lifetime of internal conversion
% 14. I_secmax : Secondary excitation intensity for modulation in
W/cm^2.
% 15. wavelength2 : wavelength of secondary laser (W/cm2)
% t_on : average on-time, t_off : average off-time

if nargin<2
    sim_mode='time';
end

I_sat = 6.6261e-34*2.9979e8/(input_param(2)*3.82356e-
21*input_param(3)*...
```

```
    10^-9*input_param(6));
%Nbins = t_mod/input_param(13);
k_exc = input_param(2)*3.82356e-21*input_param(1)*input_param(3)*10^-
9/...
    (6.6261e-34*2.9979e8*(1+input_param(1)/I_sat));

Isec = @(t)input_param(16)/2*(square(pi*t/input_param(15))+1);
kReISC = @(t)Isec(t)*input_param(15)*input_param(17)*10^-9/(6.6261e-
34*2.9979e8);

% t_ground : mean dwell time at ground state = mean waiting time of
% absorption of a photon = 1/(mean number of absorbed photons by a
molecule
% per unit time)
t_ground=1/(input_param(1)*input_param(2)*3.82356e-
21*input_param(3)*10^-9/...
    (6.6261e-34*2.9979e8));

% t_on does not depend on Fdet. Intensity and average photon waiting
time
% depend.
% k_on1 = k_exc*input_param(8,1);
% k_on2 = k_exc*input_param(8,2);
% k_off1 = 1/input_param(7,1)+kReISC(0);
% k_off2 = 1/input_param(7,2)+kReISC(0);
% Fisc1=input_param(8,1);
% Fisc2=input_param(8,2);
Ffl=input_param(4);
Fdet=input_param(5);
Fisc=input_param(8,:);
% Fic : quantum yield of internal conversion
Fic=1-sum(Fisc)-Ffl;
t_off=input_param(7,:);
t_fl=input_param(6,1);
t_ic=input_param(13,1);
t_col=input_param(12);
%input_param(6)=1/(F_dark*input_param(2)*input_param(1)*input_param(3)*
%1e-9*t_on/(input_param(4)*Fdet*k_rad*h*c)-input_param(1)*...
%    input_param(2)*input_param(3)*1e-9/(h*c));
%t_dark=1/t_off1;

kt=k_exc*input_param(11);
% kt == 1 when Iprim=h*c*I_sat/(t_bin*s_abs*wavelength*1e-9*I_sat-hc)
if input_param(12) == 0
    back_bin = 0;
else
    back_bin = input_param(10)*input_param(11);
end
if kt>1
    warning('Absorption condition is larger than 1.');
end
numStates = size(input_param,2)+4;
% state=[ground, singlet(internal conversion), singlet(photon missing),
% singlet(photon detection), dark state1, dark state2,...]
tr=zeros(numStates);
tr(1,:)=[0, Fic,Ffl*(1-Fdet),Ffl*Fdet,Fisc];
```

```
tr(2:end,1)=1;
tr=norm_m(tr);
tr(isnan(tr))=0;
% lifetime of internal conversion is very short(~fs to ~ps). If it's
not
% negligible, correct for future
lifetime=[t_ground t_ic t_fl t_fl t_off];
% allocate arrays for time duration and state for speed
if strcmp(sim_mode,'time')==1 || nargin<2
    % weight of states per dwell = p./(1./(1-p))/sum(p./(1./(1-p)));
    % weight of states per transition = p;
    % time per dwell=sum(lifetime.*weight_dwell);
    % expected number of dwell given collecting time = collectiing
time/time per dwell
    % L : expected number of state transition * 2
    pwt=sum([1,tr(1,2:end)]/2.*lifetime);
    L=round(2*t_col/(1/(1/pwt+input_param(10,1))));
    photon=zeros(round(tr(1,4)*L),1);
    % if L is too large (>1e6), memory occupation causes problem.
    % L_loop : number of iteration with smaller array.
    if L>1e6
        time_duration = zeros(1e6,numStates);
        for cc=1:numStates
            time_duration(:,cc) = exprnd(lifetime(cc),1e6,1);
        end
        % time_duration : lifetime of each state
%         states=zeros(1e6,2);
    else
        time_duration = zeros(L,numStates);
        for cc=1:numStates
            time_duration(:,cc) = exprnd(lifetime(cc),L,1);
        end
%         states=zeros(L,2);
    end
    L_loop=ceil(L/1e6);
elseif strcmp(sim_mode,'photon')==1
    % transition per photon = 1/(kt*(1-Fisc1-Fisc2)*Fdet)
    % dwell per photon = 1/(kt*(1-Fisc1-Fisc2)*Fdet)*p./(1./(1-
p))/sum(p./(1./(1-p)))
    % number of photons = input_param(14)

L=round(2*t_col/(sum(tr(1,2:numStates).*lifetime(2:numStates))+1/k_exc)
);
    % if L is too large (>1e6), memory occupation causes problem.
    % L_loop : number of iteration with smaller array.
    if L>1e6
        time_duration = zeros(1e6,numStates);
        for cc=1:numStates-1
            time_duration(:,cc) = exprnd(lifetime(cc),1e6,1);
        end
        % lifetime of photon detection is fast compared to other state.
Is
        % it possible to fix the lifetime as one????
        time_duration(:,numStates)=ones(1e6,1);
%         states=zeros(1e6,2);
    else
        time_duration = zeros(L,numStates);
```

```
%         for cc=1:5
%             time_duration(:,cc) = exprnd(lifetime(cc),L,1);
%         end
        for cc=1:numStates-1
            time_duration(:,cc) = exprnd(lifetime(cc),1e6,1);
        end
        % lifetime of photon detection is fast compared to other state.
Is
        % it possible to fix the lifetime as one????
%         states=zeros(L,2);
    end
    L_loop=ceil(L/1e6);
    photon=zeros(input_param(12),1);
else
    errors('Wrong generation mode!')
end
% ps : state of photon
ps=zeros(size(photon));
% if kt >= 1
%     warning('Absorption condition is larger than 1.');
% end

%% begin generation

% back_photon = exprnd(1/background level)
back_photon=cumsum(exprnd(1/input_param(10),ceil(2*t_col*input_param(10
)),1));
%back_photon(:,2)=state at the time backgroudn photon is detected
back_photon(:,2)=0;
if L_loop>1
    % create a random sequence for state changes
    state_random = rand(1e6,1);
else
    % create a random sequence for state changes
    state_random = rand(L,1);
end
% calculate cumulative probabilities
trc = cumsum(tr,2);
% normalize these just in case they don't sum to 1.
trc = trc./repmat(trc(:,end),1,numStates);
trc(isnan(trc))=0;
% Assume that we start in state 1.
currentstate = 1;
currenttime = time_duration(1,1);
% states(1,1)=time_duration(1,1);
state_photon=1;

% main loop
c_photon=1;
count=1;
% count_back=1;
if strcmp(sim_mode,'time')==1
    for count_loop=1:L_loop
        %if more than 1e6 state transitions are expected
%         tic
%         while currenttime<input_param(14)/t_bin && count<=1e6
```

```
        while count<=1e6 && currenttime < t_col
            % L_loop : number of iteration when L > 1e6
            % calculate state transition
            if currentstate~=1
                % After any photophysical transition, the molecule
returns
                % to ground state.
                state=1;
            else
                stateVal = state_random(count);
                state=sum(trc(currentstate,:)<=stateVal)+1;
                %find a state with cdf larger than a random number
%                 state = 1;
%                 for innerState = numStates-1:-1:1
%                     if stateVal > trc(currentstate,innerState)
%                         state = innerState + 1;
%                         break;
%                     end
%                 end
            end
%             states(count)=state;
            % add lifetime of each state
            currenttime=currenttime+time_duration(count,state);

            % write states
            if state>state_photon
                % if current state is larger tham state_photon, the
state of
                % following photon is singlet over ground, dark over
                % singlet, or another dark over dark state. In this
case,
                % update state_photon by state.
                state_photon=state;
            end
            if state==4
                % photon detection
                photon(c_photon)=currenttime;
                ps(c_photon)=state_photon;
                % update ps of photons detected during on-state : 3
                %                     photon_state=3;
                state_photon=4;
                c_photon=c_photon+1;
            end
            %update current state
            currentstate = state;
            %increase count of time duration
            count=count+1;
        end
        count=1;
        % if size of array > 1e6, generate another sequences of random
numbers
        if L_loop>1
            state_random = rand(1e6,1);
            time_duration = zeros(1e6,numStates);
            for cc=1:numStates
                time_duration(:,cc) = exprnd(lifetime(cc),1e6,1);
            end
```

```
        end
%         gen_time=toc
    end
elseif strcmp(sim_mode,'photon')
end
ps(photon==0)=[];
% ps=ps-1;
ps_ori=ps;
clear ps
photon(photon==0)=[];
%remove background photons detected after collecting time
back_photon=back_photon(back_photon(:,1)<=photon(end),:);

% ps : background photon is labeled as zero
% photon state of background photons = 0
ps=[ps_ori;zeros(length(back_photon),1)];
% index of consolidated photons
[photon,p_index]=sort([photon;back_photon(:,1)]);
ps=ps(p_index);
ps=ps-3;
ps(ps<1)=0;
% ps2 : label the state of noise photon as zero
ps2=ps;
% ps : label noise photon as hidden state when the backgrounds are
detected
dw=dwell_time(ps);
lb=find(dw(:,2)==0);
if back_bin>0
    if lb(end)==size(dw,1)
        dw(lb(1:end-1),2)=dw(lb(1:end-1)+1,2);
        dw(lb(end),2)=dw(lb(end)-1,2);
    else
        dw(lb,2)=dw(lb+1,2);
    end
    ps=dwell_time(dw,1);
end
% display('state = 1 : ground');
% display('state = 2 : singlet decay(internal conversion)');
% display('state = 3 : singlet decay(photon missing)')
% display('state = 4 : singlet decay(photon detection)')
% display('state = 5,... : dark states')
% display(' ')
% display('ps = 1 : photon detected without any transition into dark
states')
% display('ps = 2 : photon detected followed by nonzero transitions
into only dark1')
% display('ps = 3 : photon detected followed by nonzero transitions
into dark2')

%% if ps_sl is needed, uncomment a following block
% %save the state of background photons
% c_states=cumsum(states(:,1));
% for count=1:length(back_photon)
%     if sum(c_states>=back_photon(count,1))==0
%         back_photon(count,2)=states(end,2);
%     else
```

```
%
back_photon(count,2)=states(find(c_states>=back_photon(count,1),1),2);
%     end
% end
%
% % remove background photon detected at the same time of signal photon
% [C,ii]=setdiff(back_photon(:,1),photon);
% back_photon=[C,back_photon(ii,2)];
% clear C ii
%
% %ps_sl : background photon is labeled as nonzero state
% ps_sl=[ps_ori;back_photon(:,2)];
% % index of consolidated photons
% ps_sl=ps_sl(p_index);
% ps_sl(ps_sl==2)=3;
% ps_sl(ps_sl==1)=3;
% ps_sl=ps_sl-2;
% clear p_index
```

```
function macro_seq = macrotime3(raw_data, bin, chan)
% Returns a time-binned intensity trajectory from photon waiting time
% macro_seq = [time tags, counts]
% raw_data contains only photon arrival time (not photon waiting time).
% bin time is in second
% chan : the number of channels used

if size(raw_data, 1) < size(raw_data, 2)
    raw_data = raw_data';
end
if nargin<3
    chan=1;
end
%% test input data (waiting time or arrival time?)
if sum((raw_data-[0;raw_data(1:end-1)])<0)==0
    % arrival time
else
    % waiting time
    raw_data=cumsum(raw_data);
end
%% initialization
% convert time into nanosec %%
size_raw = length(raw_data);
size_macro = fix(raw_data(end) / bin) + 1;
%det_class = 1/min(raw_data-[0;raw_data(1:end-1)])*bin;
%if det_class <= 255
%    macro_seq = zeros(size_macro, chan,'uint8');
%elseif det_class <= 65535
%    macro_seq = zeros(size_macro, chan,'uint16');
%else
%    macro_seq = zeros(size_macro, chan,'uint32');
%end
%macro_seq = zeros(size_macro, chan,'uint8');
if nargin<3
    chan=1;
end
macro_seq = zeros(size_macro, chan+1);
macro_timetag = raw_data;
count = 1;
count_tbin_data = 1;

%% extract microdata from each range
while count < size_raw
    chan1 = 0;
    %chan2 = 0;
    if macro_timetag(count) < count_tbin_data*bin
        while macro_timetag(count) < count_tbin_data*bin
            chan1 = chan1 + 1;
            count = count + 1;
            if count >= size_raw
                break;
            end
        end
        %macro_seq(count_tbin_data,:) = [macro_timetag(count), chan1];
        macro_seq(count_tbin_data,2) = chan1;
%         count = count + 1;
```

```
        count_tbin_data = count_tbin_data + 1;
    else
%         while macro_timetag(count) <= count_tbin_data*bin
%             chan1 = chan1 + 1;
%             count = count + 1;
%             if count >= size_raw
%                 break;
%             end
%         end
        macro_seq(count_tbin_data,2) = chan1;
        count_tbin_data = count_tbin_data + 1;
    end
end
macro_seq(:,1)=(1:size_macro)'*bin;
```

```
function
[acf,acf2]=myac_new(seq,upper_delay,index_delay,bin_delay,lower_delay)
%%% calculate autocorrelation curve by Jung-cheng's algorithm
%%% seq : arrival times
%%% lower_delay : 1e-8(default)
%%% index_delay = 0 (log delay series)
%%% index_delay = 1 (linear delay series)

%%% correction
%%% change loop dimension : old(fix photon, change delay)

%%% reference
% Wahl, M., I. Gregor, M. Patting, and J. Enderlein,
% Fast calculation of fluorescence correlation data with asynchronous
% time-correlated single-photon counting. Optics Express, 2003. 11(26):
p. 3583-3591.

upper_dec=fix(log10(upper_delay))-1;
if nargin < 5
    lower_delay=1e-8;
    if nargin <4
        if nargin < 3
            % default delay : log-scale
            index_delay = 0;
        end
        bin_delay=1e-6;
    end
end
if upper_dec>=-7
    if index_delay==0
        %% log-delay
        delay=repmat((1:0.5:9.5)',upper_dec-
log10(lower_delay)+1,1).*...

reshape(repmat(10.^(log10(lower_delay):upper_dec),18,1),18*(upper_dec-
log10(lower_delay)+1),1);
        delay(delay>upper_delay)=[];
        if delay(end)<upper_delay
            delay=[delay;upper_delay];
        end
    elseif index_delay==1
        %% linear delay
        delay=(0:ceil(upper_delay/bin_delay))'*bin_delay;
    else
        error('Index for delay should be either one or zero.')
    end
end
% delay=cumsum(delay);
% seq1=seq;
% seq2=seq;
M=length(delay);
acf=zeros(M,2);
Na=length(seq);
[~,sort_list]=sort([seq+delay(1);seq]);
% mk1
mk1=find(sort_list<=Na)-(0:Na-1)';
```

```
for k=2:M-1
    [~,sort_list]=sort([seq+delay(k);seq]);
    mk2=find(sort_list<=Na)-(0:Na-1)';
    acf(k-1,2)=sum((mk2-mk1)/(delay(k)-delay(k-1))/(1-delay(k-
1)/seq(end)));
    % replace mk1 by mk2
    mk1=mk2;
end
acf(:,1)=delay;
acf(end,:)=[];
acf(acf(:,2)==0,:)=[];
%% acf2 : normalized and smoothed acf
acf2=acf;
acf2(:,2)=acf2(:,2)/min(acf2(:,2));
acf2(:,2)=smooth(acf2(:,2),5);
acf2=acf2(argmax(acf2(:,2)):end,:);
acf2(:,2)=acf2(:,2)/min(acf2(:,2));
```

```
function [final_fit_param,final_re_param,final_state_seq,final_fit_tr,
final_fit_time, ...
    final_pi, final_state_weight,final_cell_BIC,final_cell_chi2prob,...

final_cell_logp,final_cell_chi,final_cell_seq_logp,final_cell_seq_chi,.
..
    final_raw_e,final_raw_e_fit]= ...
    PbPHMM_findmodel(pw, s_limit,input_param,
crit_stop,crit_numst,ori_param)
% Return a photophysical model and parameters from photon waiting time
% s_limit : upper limit of the number of dark state
% input_param : input photophysical parameters and initial quantum
yields
% and lifetimes of dark states.
% crit_stop : criteria for stopping at given number of state. 'chi2' or
'logp'
% crit_numst : criteria used in determining the number of state. 'bic'
or 'chi'
% default : 'logp' for crit_stop and 'bic' for crit_numst.

% final_fit_param : estimated photophysical parameters
% final_re_param : relative errors of estimate parameters compared to
real
%                   parameters
% final_state_seq : reconstructed state sequence
% final_fit_tr : trained transition matrix
% final_fit_time : expected photon waiting time of each state
% final_pi : trained initiation matrix
% final_state_weight : population of states
% final_cell_BIC : BIC as a function of the number of state
% final_cell_chi2prob : chi-square probability as a function of the
number of state
% final_cell_logp : log-likelihood as a function of the number of state
% final_cell_chi : chi-square as a function of the number of state
% final_cell_seq_logp : log-likelihood along training at the most
probable
% number of hidden state
% final_cell_seq_chi : chi-square along training at the most probable
% number of hidden state
% final_raw_e : trained emission matrix
% final_raw_e_fit : emission matrix from trained photophysical
parameters

% correction history
% Do fitting again at given number of hidden state using extracted
% parameters as initial parameters
% index_toff : 1(use individual photons), 2(dwell time at ps)
% extract p_param in every itartion. use trained p_param for next
% iteration.
% negative adjusted r-square is allowed.
% BIC ori : inside myHMM
% BIC2 : tr, e
% BIC3 : tr, e_fit
% HMM_PbyP_back_lowdet_ranksum_meantr_3s_param_on1_newadd
% when fit at increased number of state, wait time of new state is
% geometrical average of first and second most populated states.
```

```
% state 1 is bright state.
% reason_stop
% 0 : overlap
% 1 : increasing chi2prob or decreason BIC
% 2 : zero transition probability
% 3 : zero weight  :  no access of certain state
% 4 : bad curve fitting
% 5 : infinite chi2prob  :  bad fitting
% 6 : states with the number less than 2 or more than state limit
% 7 : unknown reason
% rows of transition matrix are same.
% use initial emission matrix directly
% input : pw, init_tr, init_time
% structure of init_time
% (dw02_1,              dw02_2, dw1)
% (pwt,                  0,      0 )
% (dw3+dw1+dw02_1+dw02_2,0,      0 )
% dw02_1 : mean dwell time at state 0 or state 2 prior to transition to
% state 1
% dw02_2 : mean dwell time at state 0 or state 2 after to transition to
% state 1 before photon detection
% dw1 : mean dwell time at state 1
% pwt : mean photon waiting time at state 2
% pw : photon waiting time (not photon arrival time)

if min(pw) <= 0
    error('Minimum value of data is zero.')
end
% if sum((pw-[0;pw(1:end-1)])-abs(pw-[0;pw(1:end-1)]))~=0
%     error('pw should be photon waiting times, not photon arrival
times')
% end
%% show the histogram of data and determine the starting number of
state.
% reason_stop : why the fitting is terminated.

reason_stop=[];
if nargin < 7
    pre_fit=5;
    if nargin < 6
        ori_param=0;
    end
end

%plot(x, hist);
%grid on;
%init_level = input('Input initial guess of emissive level (ex.
[1,2,3]) : ');
st1 = size(input_param,2)+1;
% when crit is nonzero, fitting terminates.
crit = 0;
state_unit = 1;
% numE is used to calculate chi2prob.
numE=max(pw);
k_exc=absc(input_param);
back_level=input_param(10);
```

```
cell_fit_param=cell(3,1);
cell_fit_tr=cell(3,1);
cell_fit_time=cell(3,1);
cell_pi=cell(3,1);
cell_state_weight=cell(3,1);
cell_state_seq=cell(3,1);
cell_raw_e=cell(3,1);
cell_raw_e_fit=cell(3,1);
cell_seq_logp=cell(3,1);
cell_seq_chi=cell(3,1);
cell_ov=cell(3,1);
cell_global_chi = zeros(s_limit-1,1);
cell_BIC = zeros(s_limit-1,1);
cell_logp = zeros(s_limit-1,1);
cell_chi2prob = zeros(s_limit-1,1);
cell_reason_stop = cell(s_limit-1,1);
%% check photophysical relevance
if (input_param(4,1)+sum(input_param(8,:))) > 1
    error('Sum of quantum yields should be than unity.')
end
%% st1 states.

init_pi=zeros(st1,1);
init_pi(:)=1/st1;

[cell_fit_param{2},cell_fit_tr{2}, cell_fit_time{2}, cell_pi{2},
cell_seq_logp{2},...
    cell_seq_chi{2}, cell_global_chi(2), cell_state_weight{2},
cell_state_seq{2},...
    cell_raw_e{2},
cell_raw_e_fit{2},cell_reason_stop{2},final_re_param]...
    =
PbPHMM_givenstate(pw,input_param,200,init_pi,crit_stop,ori_param);

cell_logp(2)=cell_seq_logp{2}(end);

if isempty(cell_fit_tr{2})==0 && prod(isreal(cell_seq_logp{2}))~=0
    numState=st1;

    cell_chi2prob(2) = chi2cdf2(cell_global_chi(2), numE-2*(st1));
    cell_BIC(2)=BIC(cell_logp(2),length(pw),numState);
else
%     crit=crit+1;
    cell_chi2prob(2) = inf;
    cell_BIC(2)= -inf;
end
%ov_peak2 = ov_peak(cell_lambda{1,2}, cell_state_weight{1,2},
max(photon));
%% st1-1 states.
if st1 > 2


    numState=st1-1;

    if (isinf(cell_global_chi(2))==0) && (cell_logp(2)~= -inf) && ...
```

```
            (isempty(cell_fit_param{2})==0) &&
(isempty(cell_state_weight{2})==0)
        %%% if fitting at st1 states are done, remove an unoccupied
state.
        %%% Otherwise, remove the state with shorter pwt in order to
reduce
        %%% extremely small number issue in emisssion probabilities.
        [~,~,pwt]=extract_tre(cell_fit_param{2});
        [~,sort_list]=sort(pwt);
        % exclude bright state
        sort_list(sort_list==1)=[];
        index_remove=sort_list(1);
        tofffisc=cell_fit_param{2}(7:8,:);
        tofffisc(:,index_remove - 1)=[];
        input_param2=input_param(:,1:numState-1);
        input_param2(7:8,:)=tofffisc;
        init_pi=cell_pi{2};
    elseif (isempty(cell_state_weight{2})==0) &&
(prod(cell_state_weight{2})==0)
        %%% if fitting at st1 did not return results, but the state
weight
        %%% exists, find unoccupuied state
        index_remove=find(cell_state_weight{2}==0);
        init_pi=rand(st1-1,1);
        init_pi=init_pi/sum(init_pi);
        if (length(index_remove)==1) && (index_remove~=1)
            % if only one unoccupied state exists and it is not a
bright
            % state,
            tofffisc=input_param(7:8,:);
            tofffisc(:,index_remove - 1)=[];
            input_param2=input_param(:,1:numState-1);
            input_param2(7:8,:)=tofffisc;
        else
            % if multiple states are not occupied, terminate it.
            final_fit_param=[];
            final_fit_tr=[];
            final_fit_time=[];
            final_pi=[];
            final_state_weight=[];
            final_state_seq=[];
            final_cell_BIC=[];
            final_cell_chi2prob=[];
            final_cell_logp=[];
            final_cell_seq_logp=[];
            final_cell_chi = [];
            final_cell_seq_chi=[];
            final_raw_e=[];
            final_raw_e_fit=[];
            final_ov = [];
            final_re_param=[];
            reason_stop=5;
            return
        end
    else
        % if st1 fitting was not done, start from initial parameters
with
```

```
        % st1-1 states
        [~,~,pwt]=extract_tre(input_param);
        [~,sort_list]=sort(pwt);
        % exclude bright state
        sort_list(sort_list==1)=[];
        index_remove=sort_list(1);
        tofffisc=input_param(7:8,:);
        tofffisc(:,index_remove - 1)=[];
        input_param2=input_param(:,1:numState-1);
        input_param2(7:8,:)=tofffisc;
    end

    [cell_fit_param{1},cell_fit_tr{1}, cell_fit_time{1}, cell_pi{1},
cell_seq_logp{1},...
        cell_seq_chi{1}, cell_global_chi(1), cell_state_weight{1},
cell_state_seq{1},...
        cell_raw_e{1},
cell_raw_e_fit{1},cell_reason_stop{1},final_re_param]...
        =
PbPHMM_givenstate(pw,input_param2,200,init_pi,crit_stop,ori_param);

    cell_logp(1)=cell_seq_logp{1}(end);
    if isempty(cell_fit_tr{1})==0 && prod(isreal(cell_seq_logp{1}))~=0
        cell_BIC(1)=BIC(cell_logp(1),length(pw),numState);
        cell_chi2prob(1) = chi2cdf2(cell_global_chi(1), numE-2*(st1-
1));
        if prod(cell_state_weight{1}) == 0
            %if state weights with st-1 states include any zero
            cell_chi2prob(1)=inf;
            warning('One of the state weight is zero at st-1 states')
        else cell_ov{1} = ov_ranksum(pw, cell_state_seq{1});
            if isempty(cell_ov{1}) && (prod(cell_state_weight{1})==0)
&&...
                    (cell_chi2prob(1)~=Inf) && (cell_BIC(1)~=-inf)
                %if there is no overlap at st-1 states or state weights
with st
                %states include any zero,
                final_fit_param=cell_fit_param{1};
                final_fit_tr=cell_fit_tr{1};
                final_cell_seq_logp=cell_seq_logp{1};
                final_fit_time=cell_fit_time{1};
                final_pi=cell_pi{1};
                final_state_weight=cell_state_weight{1};
                final_state_seq=cell_state_seq{1};
                final_cell_BIC=cell_BIC;
                final_cell_chi2prob=cell_chi2prob;
                final_cell_logp=cell_logp;
                final_cell_chi=cell_global_chi;
                final_cell_seq_chi = cell_seq_chi{1};
                final_raw_e=cell_raw_e{1};
                final_raw_e_fit=cell_raw_e_fit{1};
                final_ov = cell_ov{1};
                reason_stop=cell_reason_stop{1};
                crit = crit + 1;
                return
            end
```

```
        end
    else
        cell_chi2prob(1)=inf;
        cell_BIC(1)=-inf;
    end
else
    cell_chi2prob(1)=inf;
    cell_BIC(1)=-inf;
end


%% st1+1 states
if crit == 0
    % add a state based on the most frequent dark state
    %     time3=zeros(st1+1,1);
    %     tr3=zeros(st1+1,1);
    %     time3(1:st1,1)=cell_fit_time{2};
    [~,sort_weight]=sort(cell_state_weight{2}(2:end));
    %     if sort_weight(end)==1
    %         add_index=sort_weight(end-1)+1;
    %     else
    %         add_index=sort_weight(end)+1;
    %     end
    %     time3(st1+1,1)=cell_fit_time{2}(add_index,:);
    %     time3(st1+1,end)=sqrt(prod(cell_fit_time{2}(sort_weight(end-
1:end)+1,1)));
    %     tr3(1:st1,1:st1)=cell_fit_tr{2};
    %     % assume the weight(quantum yield) of new state is half of
most occupied state.
    %     if cell_fit_tr{2}(1,add_index)==0
    %
tr3(:,st1+1)=min(cell_fit_tr{2}(1,cell_fit_tr{2}(2,:)>0));
    %     else
    %         tr3(:,st1+1)=cell_fit_tr{2}(1,add_index)/2;
    %     end
    %     % all trp to next state are same.
    %     tr3(st1+1,:)=tr3(1,:);
    %     tr3=norm_m(tr3);
    clear add_index
    if size(cell_pi{2},1)>size(cell_pi{2},2)
        init_pi2=[cell_pi{2};0];
    else
        init_pi2=[cell_pi{2},0];
    end
    init_pi2=init_pi2/sum(init_pi2);
    numState=st1+1;
    input_param2=zeros(size(input_param,1),numState-1);
    input_param2(:,1:numState-2)=input_param;
    % assign additional dark state quantum yield and dark-time
    if isempty(cell_fit_param{2})==0
        % if fitting at st1 states is ok, set the initial for st1+1
states
        % from st1
        input_param2(7,numState-1)=sqrt(prod(cell_fit_param{2}(7,:)));
        input_param2(8,numState-1)=sqrt(prod(cell_fit_param{2}(8,:)));
    else
```

```
        % if fitting at st1 states returns no results, set the initial
for
        % st1+1 states from the initial for st1 states.
        input_param2(7,numState-1)=sqrt(prod(input_param(7,:)));
        input_param2(8,numState-1)=sqrt(prod(input_param(8,:)));
    end

    [cell_fit_param[2],cell_fit_tr[2], cell_fit_time[2], cell_pi[2],
cell_seq_logp[2],...
        cell_seq_chi[2], cell_global_chi(3), cell_state_weight[2],
cell_state_seq[2],...
        cell_raw_e[2],
cell_raw_e_fit[2],cell_reason_stop[2],final_re_param]...
        =
PbPHMM_givenstate(pw,input_param2,200,init_pi,crit_stop,ori_param);

    cell_logp(3)=cell_seq_logp[2](end);

    if isempty(cell_fit_tr[2])==0 && prod(isreal(cell_seq_logp[2]))~=0
        cell_BIC(3)=BIC(cell_logp(3),length(pw),numState);
        cell_chi2prob(3) = chi2cdf2(cell_global_chi(3), numE-
2*(st1+1));
        if prod(cell_state_weight[2])~=0
            cell_ov[2] = ov_ranksum(pw, cell_state_seq[2]);
        end
    else
        cell_logp(3)=-Inf;
        cell_BIC(3)=-Inf;
        cell_chi2prob(3)=Inf;
        cell_ov[2] = [];
        reason_stop=cell_reason_stop{2};
        crit=crit+1;
    end
    if isempty(cell_ov[2]) == 0 || prod(cell_state_weight[2])==0 || ...
            isreal(cell_logp(3))==0
        % if overlap exists, any state is unaccessible, or log-
likelihood
        % is not a real value, mark the fitting as wrong.
        cell_logp(3)=-Inf;
        cell_BIC(3)=-Inf;
        cell_chi2prob(3)=Inf;
        cell_ov[2] = [];
        reason_stop=cell_reason_stop{2};
        crit=crit+1;
    end
else
    cell_logp(3)=-Inf;
    cell_BIC(3)=-Inf;
    cell_chi2prob(3)=Inf;
    cell_ov[2] = [];
    reason_stop=cell_reason_stop{2};
    crit=crit+1;
end
%% if fitting at st1 and st1-1 states are wrong, terminate the fitting
now
```

```
if
isinf(cell_global_chi(1))+isinf(cell_global_chi(2))+isinf(cell_global_c
hi(3)) == 3
    final_fit_param=[];
    final_fit_tr=[];
    final_fit_time=[];
    final_pi=[];
    final_state_weight=[];
    final_state_seq=[];
    final_cell_BIC=[];
    final_cell_chi2prob=[];
    final_cell_logp=[];
    final_cell_seq_logp=[];
    final_cell_chi = [];
    final_cell_seq_chi=[];
    final_raw_e=[];
    final_raw_e_fit=[];
    final_ov = [];
    final_re_param=[];
    reason_stop=5;
    display('Wrong fitting. Adjust the initial parameters')
    return
end
%% determine fitting direction

% if crit == 0
if strcmp(crit_numst,'chi')==1
    %stop
    if (cell_chi2prob(1) > cell_chi2prob(2)) && (cell_chi2prob(2) <
cell_chi2prob(3))
        reason_stop=1;
        final_fit_param=cell_fit_param{2};
        final_fit_tr=cell_fit_tr{2};
        final_fit_time=cell_fit_time{2};
        final_pi=cell_pi{2};
        final_state_weight=cell_state_weight{2};
        final_state_seq=cell_state_seq{2};
        final_cell_BIC=cell_BIC;
        final_cell_chi2prob=cell_chi2prob;
        final_cell_logp=cell_logp;
        final_cell_chi=cell_global_chi;
        final_cell_seq_chi = cell_seq_chi{2};
        final_raw_e=cell_raw_e{2};
        final_cell_seq_logp=cell_seq_logp{2};
        final_raw_e_fit=cell_raw_e_fit{2};
        final_ov = cell_ov{2};
        crit = 2;
        %backward
    elseif (cell_chi2prob(1) < cell_chi2prob(2)) && (cell_chi2prob(1) <
cell_chi2prob(3))
        state_unit = -1;
        final_fit_param=cell_fit_param{1};
        old_fit_tr=cell_fit_tr{1};
        old_fit_time=cell_fit_time{1};
        old_pi=cell_pi{1};
        old_state_weight=cell_state_weight{1};
        old_state_seq=cell_state_seq{1};
```

```
        old_chi2prob=cell_chi2prob(1);
        old_seq_logp=cell_seq_logp{1};
        old_seq_chi=cell_seq_chi{1};
        old_raw_e=cell_raw_e{1};
        old_raw_e_fit=cell_raw_e_fit{1};
        old_ov = cell_ov{1};
        old_reason_stop=cell_reason_stop{1};
        numState = st1-1;
        if st1 == 3
            reason_stop=6;
            final_fit_param=cell_fit_param{1};
            final_fit_tr=cell_fit_tr{1};
            final_fit_time=cell_fit_time{1};
            final_pi=cell_pi{1};
            final_state_weight=cell_state_weight{1};
            final_state_seq=cell_state_seq{1};
            final_cell_BIC=cell_BIC;
            final_cell_chi2prob=cell_chi2prob;
            final_cell_logp=cell_logp;
            final_cell_seq_logp=cell_seq_logp{1};
            final_cell_chi=cell_global_chi;
            final_cell_seq_chi = cell_seq_chi{1};
            final_raw_e=cell_raw_e{1};
            final_raw_e_fit=cell_raw_e_fit{1};
            final_ov = cell_ov{1};
            crit = crit + 2;
            return
        end
        %forward
    elseif (cell_chi2prob(3) <= cell_chi2prob(1)) && (cell_chi2prob(3)
<= cell_chi2prob(2))
        state_unit = 1;
        old_fit_param=cell_fit_param[2];
        old_fit_tr=cell_fit_tr[2];
        old_fit_time=cell_fit_time[2];
        old_pi=cell_pi[2];
        old_state_weight=cell_state_weight[2];
        old_state_seq=cell_state_seq[2];
        old_seq_logp=cell_seq_logp[2];
        old_seq_chi=cell_seq_chi[2];
        old_chi2prob=cell_chi2prob(3);
        old_raw_e=cell_raw_e[2];
        old_raw_e_fit=cell_raw_e_fit[2];
        old_ov = cell_ov[2];
        old_reason_stop=cell_reason_stop[2];
        numState = st1+1;
    end
elseif strcmp(crit_numst,'bic')==1 || strcmp(crit_numst,'BIC')==1
    if (cell_BIC(1) < cell_BIC(2)) && (cell_BIC(2) > cell_BIC(3))
        reason_stop=1;
        final_fit_param=cell_fit_param{2};
        final_fit_tr=cell_fit_tr{2};
        final_fit_time=cell_fit_time{2};
        final_pi=cell_pi{2};
        final_state_weight=cell_state_weight{2};
        final_state_seq=cell_state_seq{2};
        final_cell_BIC=cell_BIC;
```

```
    final_cell_chi2prob=cell_chi2prob;
    final_cell_logp=cell_logp;
    final_cell_chi=cell_global_chi;
    final_cell_seq_logp=cell_seq_logp{2};
    final_cell_seq_chi = cell_seq_chi{2};
    final_raw_e=cell_raw_e{2};
    final_raw_e_fit=cell_raw_e_fit{2};
    final_ov = cell_ov{2};
    crit = 2;
    return
    %backward
elseif (cell_BIC(1) > cell_BIC(2)) && (cell_BIC(1) > cell_BIC(3))
    state_unit = -1;
    final_fit_param=cell_fit_param{1};
    old_fit_tr=cell_fit_tr{1};
    old_fit_time=cell_fit_time{1};
    old_pi=cell_pi{1};
    old_state_weight=cell_state_weight{1};
    old_state_seq=cell_state_seq{1};
    old_chi2prob=cell_chi2prob(1);
    old_seq_logp=cell_seq_logp{1};
    old_seq_chi=cell_seq_chi{1};
    old_raw_e=cell_raw_e{1};
    old_raw_e_fit=cell_raw_e_fit{1};
    old_ov = cell_ov{1};
    old_BIC=cell_BIC(1);
    old_reason_stop=cell_reason_stop{1};
    numState = st1-1;
    if st1 == 3
        reason_stop=0;
        final_fit_param=cell_fit_param{1};
        final_fit_tr=cell_fit_tr{1};
        final_fit_time=cell_fit_time{1};
        final_pi=cell_pi{1};
        final_state_weight=cell_state_weight{1};
        final_state_seq=cell_state_seq{1};
        final_cell_BIC=cell_BIC;
        final_cell_chi2prob=cell_chi2prob;
        final_cell_logp=cell_logp;
        final_cell_chi=cell_global_chi;
        final_cell_seq_logp=cell_seq_logp{1};
        final_cell_seq_chi = cell_seq_chi{1};
        final_raw_e=cell_raw_e{1};
        final_raw_e_fit=cell_raw_e_fit{1};
        final_ov = cell_ov{1};
        crit = crit + 2;
        return
    end
    %forward
elseif (cell_BIC(3) >= cell_BIC(1)) && (cell_BIC(3) >= cell_BIC(2))
    state_unit = 1;
    old_fit_param=cell_fit_param[2];
    old_fit_tr=cell_fit_tr[2];
    old_fit_time=cell_fit_time[2];
    old_pi=cell_pi[2];
    old_state_weight=cell_state_weight[2];
    old_state_seq=cell_state_seq[2];
```

```
        old_chi2prob=cell_chi2prob(3);
        old_seq_logp=cell_seq_logp[2];
        old_seq_chi=cell_seq_chi[2];
        old_raw_e=cell_raw_e[2];
        old_raw_e_fit=cell_raw_e_fit[2];
        old_ov = cell_ov[2];
        old_BIC=cell_BIC(3);
        old_reason_stop=cell_reason_stop[2];
        numState = st1+1;
    end
end
%% start the overall iteration
%iter : the number of iteration
iter = 1;
while crit == 0
    if state_unit == -1
        numState = numState - 1;
        new_tr=old_fit_tr;
        new_tr(argmin(old_state_weight),:)=[];
        new_tr(:,argmin(old_state_weight))=[];
        new_tr=norm_m(new_tr);
        new_time=old_fit_time;
        new_time(argmin(old_state_weight),:)=[];
        numState = numState - 1;
        init_pi2(argmin(init_pi2))=[];
        init_pi2=init_pi2/sum(init_pi2);
    elseif state_unit == 1
        pw(1)
        numState = numState + 1;
        [~,sort_weight]=sort(old_state_weight(2:end));
        if size(old_pi,2)>size(old_pi,1)
            old_pi=old_pi';
        end
        init_pi2=[old_pi;0];
        init_pi2=init_pi2/sum(init_pi2);
        init_param=zeros(size(old_fit_param,1),numState-1);
        init_param(:,1:numState-2)=old_fit_param;

        init_param(7,numState-
1)=sqrt(prod(old_fit_time(sort_weight(end-1:end)+1)));
        init_param(8,numState-1)=sqrt(prod(old_fit_param(8,1:numState-
2)));
        %         init_param(8,numState-1)=(1-sum(init_param(8,:)))/5;
    end

    [new_fit_param,new_fit_tr, new_fit_time, new_pi, new_seq_logp,...
        new_seq_chi, new_global_chi, new_state_weight,
new_state_seq,...
        new_raw_e, new_raw_e_fit,new_reason_stop,final_re_param]...
        = PbPHMM_givenstate(pw,init_param,200,pi,crit_stop,ori_param);
    new_logp=new_seq_logp(end);

    if exist('new_fit_tr','var') && (prod(isreal(new_seq_logp))~=0) &&
(isempty(new_fit_param)==0)
        % Stop fitting and return old_results if new_fit_tr does not
exist
```

```
        % due to error or complex likelihood appears.

        cell_logp(3+iter) = new_logp;
        cell_global_chi(3+iter) = new_global_chi;

        %         [~,new_logp2(3+iter)] =
hmmdecode_sb(pw,new_fit_tr,new_raw_e_fit,...
        %
1./(1./sum(new_fit_time,2)/t_bin+back_level)/t_bin,new_pi);
        cell_BIC(3+iter)=BIC(cell_logp(3+iter),length(pw),numState);
        %         cell_BIC3(3+iter) = new_logp2(3+iter) - (st1^2+st1-
1)*log(length(pw))/2;

        cell_chi2prob(3+iter) = chi2cdf2(new_global_chi, numE-
2*(st1+iter+1));
        new_chi2prob = cell_chi2prob(3+iter);
        new_BIC=cell_BIC(3+iter);
        %numState = numState + state_unit;
        iter=iter+1;
        % if any weight is zero, terminate
        % if all weight are nonzero, do ranksum test
        if prod(new_state_weight)~=0
            new_ov = ov_ranksum(pw, new_state_seq);
        else
            reason_stop=[reason_stop,2];
            final_fit_param=old_fit_param;
            final_fit_tr=old_fit_tr;
            final_fit_time=old_fit_time;
            final_pi=old_pi;
            final_state_weight=old_state_weight;
            final_state_seq=old_state_seq;
            final_cell_BIC=cell_BIC;
            final_cell_chi2prob=cell_chi2prob;
            final_cell_logp=cell_logp;
            final_cell_chi=cell_global_chi;
            final_cell_seq_logp=old_seq_logp;
            final_cell_seq_chi = old_seq_chi;
            final_raw_e=old_raw_e;
            final_raw_e_fit=old_raw_e_fit;
            reason_stop=old_reason_stop;
            final_ov=[];
            crit = crit + 2;
            return
        end
    else
        % null result of new_fit_tr : bad fitting
        reason_stop=[reason_stop,2];
        final_fit_param=old_fit_param;
        final_fit_tr=old_fit_tr;
        final_fit_time=old_fit_time;
        final_pi=old_pi;
        final_state_weight=old_state_weight;
        final_state_seq=old_state_seq;
        final_cell_BIC=cell_BIC;
        final_cell_chi2prob=cell_chi2prob;
        final_cell_logp=cell_logp;
```

```
    final_cell_chi=cell_global_chi;
    final_cell_seq_logp=old_seq_logp;
    final_cell_seq_chi = old_seq_chi;
    final_raw_e=old_raw_e;
    final_raw_e_fit=old_raw_e_fit;
    reason_stop=old_reason_stop;
    final_ov=[];
    crit = crit + 2;
    return
end
if strcmp(crit_numst,'chi')==1
    if new_chi2prob < old_chi2prob && numState == 2
        reason_stop=0;
        final_fit_param=new_fit_param;
        final_fit_tr=new_fit_tr;
        final_fit_time=new_fit_time;
        final_pi=new_pi;
        final_state_weight=new_state_weight;
        final_state_seq=new_state_seq;
        final_cell_BIC=cell_BIC;
        final_cell_chi2prob=cell_chi2prob;
        final_cell_logp=cell_logp;
        final_cell_chi=cell_global_chi;
        final_cell_seq_logp=new_seq_logp;
        final_cell_seq_chi = new_seq_chi;
        final_raw_e=new_raw_e;
        final_raw_e_fit=new_raw_e_fit;
        final_ov=new_ov;
        crit = crit + 2;
        return
    end
elseif strcmp(crit_numst,'BIC')==1 || strcmp(crit_numst,'bic')==1
    if new_BIC < old_BIC && numState == 2
        reason_stop=0;
        final_fit_param=new_fit_param;
        final_fit_tr=new_fit_tr;
        final_fit_time=new_fit_time;
        final_pi=new_pi;
        final_state_weight=new_state_weight;
        final_state_seq=new_state_seq;
        final_cell_BIC=cell_BIC;
        final_cell_chi2prob=cell_chi2prob;
        final_cell_logp=cell_logp;
        final_cell_chi=cell_global_chi;
        final_cell_seq_logp=new_seq_logp;
        final_cell_seq_chi = new_seq_chi;
        final_raw_e=new_raw_e;
        final_raw_e_fit=new_raw_e_fit;
        %             final_re_param=old_re_param;
        final_ov=new_ov;
        crit = crit + 2;
        return
    end
end
if (numState == 1 || numState > s_limit) && (crit==0)
    reason_stop=6;
    final_fit_param=old_fit_param;
```

```
    final_fit_tr=old_fit_tr;
    final_fit_time=old_fit_time;
    final_pi=old_pi;
    final_state_weight=old_state_weight;
    final_state_seq=old_state_seq;
    final_cell_BIC=cell_BIC;
    final_cell_chi2prob=cell_chi2prob;
    final_cell_logp=cell_logp;
    final_cell_seq_logp=old_seq_logp;
    final_cell_chi = cell_global_chi;
    final_cell_seq_chi = new_seq_chi;
    final_raw_e=old_raw_e;
    final_raw_e_fit=old_raw_e_fit;
    %         final_re_param=old_re_param;
    final_ov=old_ov;
    reason_stop=old_reason_stop;
    crit = crit + 2;
    return
end

if strcmp(crit_numst,'chi')==1
    if isempty(new_ov) == 0 && crit==0
        final_fit_param=old_fit_param;
        final_fit_tr=old_fit_tr;
        final_fit_time=old_fit_time;
        final_pi=old_pi;
        final_state_weight=old_state_weight;
        final_state_seq=old_state_seq;
        final_raw_e=old_raw_e;
        final_raw_e_fit=old_raw_e_fit;
        final_cell_BIC=cell_BIC;
        final_cell_chi2prob=cell_chi2prob;
        final_cell_logp=cell_logp;
        final_cell_seq_logp=old_seq_logp;
        final_cell_seq_chi = old_seq_chi;
        final_cell_chi = cell_global_chi;
        final_ov=new_ov;
        %             final_re_param=old_re_param;
        reason_stop=[new_reason_stop,0];
        crit = crit + 1;
        return
    end
    if new_chi2prob < old_chi2prob
        old_chi2prob = new_chi2prob;
        old_fit_param=new_fit_param;
        old_fit_tr=new_fit_tr;
        old_fit_time=new_fit_time;
        old_pi=new_pi;
        old_state_weight=new_state_weight;
        old_state_seq=new_state_seq;
        old_seq_logp=new_seq_logp;
        old_seq_chi=new_seq_chi;
        old_raw_e=new_raw_e;
        old_raw_e_fit=new_raw_e_fit;
        old_ov=new_ov;
        old_reason_stop=new_reason_stop;
    elseif new_chi2prob > old_chi2prob
```

```
            % if new_chi2prob > old_chi2prob, do another fitting with
            % increased(or decreased) number of hidden states.
            if state_unit == -1
                numState = numState - 1;
                new_tr=new_fit_tr;
                new_tr(argmin(new_state_weight),:)=[];
                new_tr(:,argmin(new_state_weight))=[];
                new_tr=norm_m(new_tr);
                new_time=new_fit_time;
                new_time(argmin(cell_state_weight{2}),:)=[];
            elseif state_unit == 1
                numState = numState + 1;
                %%% In these iterations, only p_params are modified.
                %%% Modifications of tr and time will be removed.
                [~,sort_weight]=sort(new_state_weight);

                clear add_index
                if size(new_pi,2)>size(new_pi,1)
                    new_pi=new_pi';
                end
                init_pi2=[new_pi;0];
                init_pi2=init_pi2/sum(init_pi2);
                init_param=zeros(15,numState-1);
                init_param(:,1:numState-2)=new_fit_param;
                %                 init_param(7,numState-
1)=init_param(7,numState-2)/2;
                init_param(7,numState-
1)=sqrt(prod(new_fit_param(sort_weight(end-1:end)+1)));
                %                 init_param(8,numState-
1)=init_param(8,numState-2)*10;
                init_param(8,numState-
1)=sqrt(prod(new_fit_param(8,1:numState-2)));
                %                 init_param(8,numState-1)=(1-
sum(init_param(8,:)))/2;
            end

            [next_fit_param,next_fit_tr, next_fit_time, next_pi,
next_seq_logp,...
                next_seq_chi, next_global_chi, next_state_weight,
next_state_seq,...
                next_raw_e,
next_raw_e_fit,next_reason_stop,final_re_param]...
                =
PbPHMM_givenstate(pw,init_param,200,pi,crit_stop,ori_param);
            next_logp=next_seq_logp(end);

            if exist('next_fit_tr','var') &&
(prod(isreal(next_seq_logp))~=0) && (isempty(next_fit_param)==0)
                % Stop fitting and return old_results if new_fit_tr
does not exist
                % due to error or complex likelihood appears.
                cell_logp(3+iter) = max(next_logp);
                cell_global_chi(3+iter) = next_global_chi;

                %                 [~,new_logp2(3+iter)] =
hmmdecode_sb(pw,next_fit_tr,next_raw_e_fit,...
```

```
                %
1./(1./sum(next_fit_time,2)/t_bin+back_level)/t_bin,next_pi);

cell_BIC(3+iter)=BIC(cell_logp(3+iter),length(pw),numState);
                %                 cell_BIC3(3+iter) = new_logp2(3+iter)
- (st1^2+st1-1)*log(length(pw))/2;
                %                 cell_BIC(3+iter)=cell_BIC3(3+iter);
                %                 cell_logp(3+iter)=new_logp2(3+iter);

                cell_chi2prob(3+iter) = chi2cdf2(next_global_chi, numE-
2*(st1+iter+1));
                next_chi2prob = cell_chi2prob(3+iter);
                next_BIC=cell_BIC(3+iter);
            else
                reason_stop=[old_reason_stop,2];
                final_fit_param=old_fit_param;
                final_fit_tr=old_fit_tr;
                final_fit_time=old_fit_time;
                final_pi=old_pi;
                final_state_weight=old_state_weight;
                final_state_seq=old_state_seq;
                final_cell_BIC=cell_BIC;
                final_cell_chi2prob=cell_chi2prob;
                final_cell_logp=cell_logp;
                final_cell_chi = cell_global_chi;
                final_cell_seq_logp=old_seq_logp;
                final_cell_seq_chi=old_seq_chi;
                final_raw_e=old_raw_e;
                final_raw_e_fit=old_raw_e_fit;
                %                 final_re_param=old_re_param;
                final_ov=[];
                return
            end
            iter=iter+1;

            % if any weight is zero, terminate
            % if all weight are nonzero, do ranksum test
            if prod(next_state_weight)~=0
                next_ov = ov_ranksum(pw, new_state_seq);
            else
                reason_stop=[old_reason_stop,2];
                final_fit_param=old_fit_param;
                final_fit_tr=old_fit_tr;
                final_fit_time=old_fit_time;
                final_pi=old_pi;
                final_state_weight=old_state_weight;
                final_state_seq=old_state_seq;
                final_cell_BIC=cell_BIC;
                final_cell_chi2prob=cell_chi2prob;
                final_cell_logp=cell_logp;
                final_cell_chi = cell_global_chi;
                final_cell_seq_logp=old_seq_logp;
                final_cell_seq_chi = old_seq_chi;
                final_raw_e=old_raw_e;
                final_raw_e_fit=old_raw_e_fit;
                %                 final_re_param=old_re_param;
```

```
    final_ov=[];
    return
end
% nonzero overlap, terminate
if isempty(next_ov) == 0 && crit == 0
    reason_stop=[reason_stop,0];
    final_fit_param=old_fit_param;
    final_fit_tr=old_fit_tr;
    final_fit_time=old_fit_time;
    final_pi=old_pi;
    final_state_weight=old_state_weight;
    final_state_seq=old_state_seq;
    final_cell_BIC=cell_BIC;
    final_cell_chi2prob=cell_chi2prob;
    final_cell_logp=cell_logp;
    final_cell_chi = cell_global_chi;
    final_cell_seq_logp=old_seq_logp;
    final_cell_seq_chi = old_seq_chi;
    final_raw_e=old_raw_e;
    final_raw_e_fit=old_raw_e_fit;
    %                 final_re_param=old_re_param;
    final_ov=old_ov;
    reason_stop=[old_reason_stop,0];
    return
    crit = crit + 1;
end

% continue fitting
if next_chi2prob < old_chi2prob && crit == 0
    old_chi2prob=next_chi2prob;
    old_fit_param=next_fit_param;
    old_fit_tr=next_fit_tr;
    old_fit_time=next_fit_time;
    old_pi=next_pi;
    old_state_weight=next_state_weight;
    old_state_seq=next_state_seq;
    old_seq_logp=next_seq_logp;
    old_seq_chi=next_seq_chi;
    old_raw_e=next_raw_e;
    old_raw_e_fit=next_raw_e_fit;
    old_reason_stop=next_reason_stop;
    %                 old_re_param=next_re_param;
elseif next_chi2prob >= old_chi2prob && crit == 0
    reason_stop=[reason_stop,1];
    final_fit_param=old_fit_param;
    final_fit_tr=old_fit_tr;
    final_fit_time=old_fit_time;
    final_pi=old_pi;
    final_state_weight=old_state_weight;
    final_state_seq=old_state_seq;
    final_cell_BIC=cell_BIC;
    final_cell_chi2prob=cell_chi2prob;
    final_cell_logp=cell_logp;
    final_cell_chi = cell_global_chi;
    final_cell_seq_logp=old_seq_logp;
    final_cell_seq_chi = old_seq_chi;
    final_raw_e=old_raw_e;
```

```
                %                 final_re_param=old_re_param;
                final_raw_e_fit=old_raw_e_fit;
                final_ov=old_ov;
                return
                crit = crit + 1;
            end
        end
    elseif strcmp(crit_numst,'BIC')==1 || strcmp(crit_numst,'bic')==1
        if new_BIC > old_BIC
            old_BIC=new_BIC;
            old_fit_param=new_fit_param;
            old_fit_tr=new_fit_tr;
            old_fit_time=new_fit_time;
            old_pi=new_pi;
            old_state_weight=new_state_weight;
            old_state_seq=new_state_seq;
            old_seq_logp=new_seq_logp;
            old_seq_chi=new_seq_chi;
            old_raw_e=new_raw_e;
            old_raw_e_fit=new_raw_e_fit;
            old_ov=new_ov;
            %             old_re_param=new_re_param;
            old_reason_stop=new_reason_stop;

        elseif new_BIC < old_BIC
            %        numState
            if state_unit == -1
                numState = numState - 1;
                new_tr=new_fit_tr;
                new_tr(argmin(new_state_weight),:)=[];
                new_tr(:,argmin(new_state_weight))=[];
                new_tr=norm_m(new_tr);
                new_time=new_fit_time;
                new_time(argmin(cell_state_weight{2}),:)=[];
            elseif state_unit == 1
                numState = numState + 1;
                [~,sort_weight]=sort(new_state_weight);
                if size(new_pi,2)>size(new_pi,1)
                    new_pi=new_pi';
                end
                init_pi2=[new_pi;0];
                init_pi2=init_pi2/sum(init_pi2);
                init_param=zeros(15,numState-1);
                init_param(:,1:numState-2)=new_fit_param;
                init_param(7,numState-
1)=sqrt(prod(new_fit_time(sort_weight(end-1:end)+0)));
                %                 init_param(8,numState-1)=(1-
sum(init_param(8,:)))/2;
                init_param(8,numState-
1)=sqrt(prod(new_fit_param(8,1:numState-2)));
            end

            [next_fit_param,next_fit_tr, next_fit_time, next_pi,
next_seq_logp,...
                next_seq_chi, next_global_chi, next_state_weight,
next_state_seq,...
```

```
                next_raw_e,
next_raw_e_fit,next_reason_stop,final_re_param]...
                =
PbPHMM_givenstate(pw,init_param,200,pi,crit_stop,ori_param);

            next_logp=next_seq_logp(end);

            if exist('next_fit_tr','var') &&
(prod(isreal(next_seq_logp))~=0) && (isempty(next_fit_param)==0)
                %
next_fit_param=extract_p_param_new_deconv_num_new_correct_newest(pw,nex
t_state_seq,...
                %
next_fit_tr,next_raw_e_fit,input_param,0);
                cell_logp(3+iter) = next_logp;
                cell_global_chi(3+iter) = next_global_chi;

cell_BIC(3+iter)=BIC(cell_logp(3+iter),length(pw),numState);

                cell_chi2prob(3+iter) = chi2cdf2(next_global_chi, numE-
2*(st1+iter+1));
                next_chi2prob = cell_chi2prob(3+iter);
                next_BIC=cell_BIC(3+iter);
                iter=iter+1;
            else
                reason_stop=[old_reason_stop,2];
                final_fit_param=old_fit_param;
                final_fit_tr=old_fit_tr;
                final_fit_time=old_fit_time;
                final_pi=old_pi;
                final_state_weight=old_state_weight;
                final_state_seq=old_state_seq;
                final_cell_BIC=cell_BIC;
                final_cell_chi2prob=cell_chi2prob;
                final_cell_logp=cell_logp;
                final_cell_chi = cell_global_chi;
                final_cell_seq_logp=old_seq_logp;
                final_cell_seq_chi = old_seq_chi;
                final_raw_e=old_raw_e;
                final_raw_e_fit=old_raw_e_fit;
                %                 final_re_param=old_re_param;
                final_ov=[];
                crit = crit + 2;
                return
            end
            % if any weight is zero, terminate
            % if all weight are nonzero, do ranksum test
            if prod(next_state_weight)~=0
                next_ov = ov_ranksum(pw, new_state_seq);
            else
                reason_stop=[old_reason_stop,2];
                final_fit_param=old_fit_param;
                final_fit_tr=old_fit_tr;
                final_fit_time=old_fit_time;
                final_pi=old_pi;
                final_state_weight=old_state_weight;
```

```
    final_state_seq=old_state_seq;
    final_cell_BIC=cell_BIC;
    final_cell_chi2prob=cell_chi2prob;
    final_cell_logp=cell_logp;
    final_cell_chi = cell_global_chi;
    final_cell_seq_logp=old_seq_logp;
    final_cell_seq_chi = old_seq_chi;
    final_raw_e=old_raw_e;
    final_raw_e_fit=old_raw_e_fit;
    %                 final_re_param=old_re_param;
    final_ov=[];
    crit = crit + 2;
    return
end
% nonzero overlap, terminate
if isempty(next_ov) == 0 && crit == 0
    reason_stop=[reason_stop,0];
    final_fit_param=old_fit_param;
    final_fit_tr=old_fit_tr;
    final_fit_time=old_fit_time;
    final_pi=old_pi;
    final_state_weight=old_state_weight;
    final_state_seq=old_state_seq;
    final_cell_BIC=cell_BIC;
    final_cell_chi2prob=cell_chi2prob;
    final_cell_logp=cell_logp;
    final_cell_chi = cell_global_chi;
    final_cell_seq_logp=old_seq_logp;
    final_cell_seq_chi = old_seq_chi;
    %                 final_re_param=old_re_param;
    final_raw_e=old_raw_e;
    final_raw_e_fit=old_raw_e_fit;
    final_ov=old_ov;
    crit = crit + 1;
    return
end

%not sure whether
if (next_BIC > old_BIC) && (crit == 0)
    old_BIC=next_BIC;
    old_fit_param=next_fit_param;
    old_fit_tr=next_fit_tr;
    old_fit_time=next_fit_time;
    old_pi=next_pi;
    old_state_weight=next_state_weight;
    old_state_seq=next_state_seq;
    old_raw_e=next_raw_e;
    old_seq_logp=next_seq_logp;
    old_seq_chi=next_seq_chi;
    old_raw_e_fit=next_raw_e_fit;
    %                 old_re_param=next_re_param;
    old_reason_stop=next_reason_stop;
elseif (next_BIC <= old_BIC) && (crit == 0)
    reason_stop=[reason_stop,1];
    final_fit_param=old_fit_param;
    final_fit_tr=old_fit_tr;
    final_fit_time=old_fit_time;
```

```
                final_pi=old_pi;
                final_state_weight=old_state_weight;
                final_state_seq=old_state_seq;
                final_cell_BIC=cell_BIC;
                final_cell_chi2prob=cell_chi2prob;
                final_cell_logp=cell_logp;
                final_cell_chi = cell_global_chi;
                final_cell_seq_logp=old_seq_logp;
                final_cell_seq_chi = old_seq_chi;
                final_raw_e=old_raw_e;
                final_raw_e_fit=old_raw_e_fit;
                %                 final_re_param=old_re_param;
                final_ov=old_ov;
                return
                %                 crit = crit + 1;
            end
        end
    end
end
%
final_fit_param=extract_p_param_new_deconv_num_new_correct_newest(pw,fi
nal_state_seq,...
%     final_fit_tr,final_raw_e_fit,input_param,1);
```

```
function [final_fit_param,final_fit_tr, final_fit_time, final_pi,
final_seq_logp,...
    final_seq_global_chi, final_global_chi, final_state_weight,
final_state_seq,...
    final_raw_e, final_raw_e_fit,reason_stop,re_param]...
    = PbPHMM_givenstate(seq,input_param, n_iter,...
    init_pi,crit_stop,ori_param)
% Train transition and emission matrix with photophysical constraint at
the
% given number of hidden state

% Repeat PbPHMM_unit until a stopping criterion is reached, keeping
% photophysical relevance


% seq : binned photon waiting time
% input_param : initial photophysical parameters
% n_iter : upper limit of iterations
% init_pi : initial initiation probabilities
% crit_stop : criteria for stopping iteration.
%   'logp'(default) : maximum log-likelihood
%   'chi' : minimum chi-square between trained emission matrix and e
from
%   trained photophysical parameters

% final_fit_param : estimated photophysical parameters
% final_state_seq : reconstructed state sequence
% final_fit_tr : trained transition matrix
% final_fit_time : expected photon waiting time of each state
% final_pi : trained initiation matrix
% final_state_weight : population of states
% final_seq_global_chi : log-likelihood along training at the most
probable
% number of hidden state
% final_raw_e : trained emission matrix
% final_raw_e_fit : emission matrix from trained photophysical
parameters
% reason_stop : cause of stopping iteration (need to be refined)
% re_param : relative errors of estimate parameters compared to real
%                   parameters


%%% correction history
% new re_param : used log instead of subtracting real values
% all state populations should be positive integers.
% relevance of fit_param is not checked again after HMM_unit returns
it.
% likelihood(logp) or chi-square(chi2) can be used as stopping point
% after an iteration by BW, photophysical parameters are extracted
once.
% need to be
%
'myHMM_waiting_raw_faster_num_logp_piter_rawe_auto_pfit_trseq_newchi',
% but the filename cannot be too long.
% fit photophysical parameters every iteration. Use trained params for
next
```

```
% iteration.
% chi-square between raw_e and raw_e_fit from extract_tre
% prepare photophysical parameters
% index_toff : 0(use individual photons), nonzero(dwell time at ps)
% fix index_toff as zero
% Fdet=input_param(5);
% Ffl=input_param(4);
% Fcol=Fdet*Ffl;
% toff1=input_param(7,1);
% Fisc=sum(input_param(8,:));
% I_prim=input_param(1);
% s_abs=input_param(2)*3.82356e-21;
% wavelength=input_param(3);
% t_fl=input_param(6);
% t_ic=input_param(13,1);
t_bin = input_param(11);
% back_level=input_param(10);
% I_sat = 6.6261e-34*2.9979e8/(s_abs*wavelength*10^-9*t_fl);
%Nbins = t_mod/tbin;

%% initialize and first fit

%%% determine seq is binned or raw
if sum(abs(seq-round(seq)))~=0
    % raw
    seq=ceil(seq/t_bin);
else
end

% count : number iteration

c_fit=1;
numState = size(input_param, 2)+1;
if sum(abs(seq-round(seq)))~=0
    numEmis = ceil(max(seq)/t_bin);
else
    numEmis = max(seq);
end
index = 0;
final_seq_logp = zeros(100,1);
final_seq_global_chi = zeros(100,1);


if nargin == 6
    re_param=zeros(100,2*(numState-1));
end

% tr and e are not initially assigned.
% tr and e are generated from input_param
[init_tr,init_e,init_time]=extract_tre(input_param,numEmis);

init_time=ceil(init_time/t_bin);
if nargin<6
    ori_param=0;
    if nargin<5
```

```
        crit_stop = 'logp';
        if nargin < 4
            init_pi=zeros(numState,1);
            if nargin < 2
                % if nargin is not determined, set the number of
iteration is 15.
                n_iter=15;
            end
        end
    end
end
if seq(1)<6
    init_pi(2)=1;
else
    init_pi(:,1) = 1/numState;
end

if size(init_pi, 1) < size (init_pi,2)
    init_pi=init_pi';
end

% [init_tr,init_e]=extract_tre(input_param,numEmis);

% tic
[old_fit_param,old_fit_tr, old_fit_time, old_pi, old_global_chi,
old_state_seq,...
    old_state_weight, old_logp, old_raw_e, old_raw_e_fit,reason_stop] =
...
    PbPHMM_unit(seq, init_tr,init_e, init_time, input_param,init_pi);
% fp(c_fit,:)=old_fit_param(7:8)';
% trp(c_fit,:)=old_fit_tr(1,:);

% fp_ori=[ori_param(7,:),ori_param(8,:)];
if isempty(old_raw_e)==0
    % if old_fit_param is ok
    if nargin>=6 && (sum(abs(size(ori_param)-size(old_fit_param)))==0)
%         re_param(c_fit,:)=[(old_fit_param(7,:)-
ori_param(7,:))./ori_param(7,:),...
%             (old_fit_param(8,:)-ori_param(8,:))./ori_param(8,:)]*100;
%
re_param(c_fit,:)=log([old_fit_param(7,:),old_fit_param(8,:)]...
%             ./fp_ori)/log(2)*100;
        re_param(c_fit,:)=re_error(old_fit_param,ori_param);
    end
end

final_seq_logp(c_fit) = old_logp;
final_seq_global_chi(c_fit) = old_global_chi;

if (prod(sum(old_fit_tr,1))*prod(sum(old_fit_tr,2))==0) || ...
        isinf(old_global_chi(1)) || (prod(old_state_weight)==0) || ...
        prod(sum(old_fit_time,2))==0
    final_fit_param=[];
    final_fit_tr=[];
    final_fit_time=[];
    final_pi=[];
```

```
    final_seq_logp=-inf;
    final_seq_global_chi=[];
    final_global_chi=Inf;
    final_state_weight=old_state_weight;
    final_state_seq=[];
    final_raw_e=[];
    final_raw_e_fit=[];
    re_param=[];
    return
end

%% iteratons
if isempty(old_fit_tr)==0
    while c_fit<n_iter

        [init_tr,init_e,init_time]=extract_tre(old_fit_param,numEmis);
        c_fit=c_fit+1;
%         try
            [new_fit_param,new_fit_tr, new_fit_time, new_pi,
new_global_chi, new_state_seq,...
                new_state_weight, new_logp, new_raw_e, new_raw_e_fit,
reason_stop] = ...
                PbPHMM_unit(seq, init_tr, init_e, init_time,
input_param,old_pi);
%             fp(c_fit,:)=new_fit_param(7:8)';
%             trp(c_fit,:)=new_fit_tr(1,:);
%         catch err
%             err
%             %             c_fit
%         end
        if isempty(new_raw_e)==0
            if nargin>=6 && (sum(abs(size(ori_param)-
size(new_fit_param)))==0)
                re_param(c_fit,:)=re_error(new_fit_param,ori_param);
            end
        end

        final_seq_logp(c_fit) = new_logp;
        final_seq_global_chi(c_fit) = new_global_chi(1);

        if (prod(sum(new_fit_tr,1))*prod(sum(new_fit_tr,2))==0) || ...
                isinf(new_global_chi(1)) ||
(prod(sum(new_fit_time,2))==0)...
                || (prod(new_state_weight)==0)
            % Viterbi algorithm has problem at small pwt misassigning
            % photons at dark state as photons from bright transitions.
            % Therefore ignore checking zero weights.
            %                 || (prod(new_state_weight)==0)
            final_fit_param=old_fit_param;
            final_fit_tr=old_fit_tr;
            final_raw_e = old_raw_e;
            final_raw_e_fit = old_raw_e_fit;
            final_pi = old_pi;
            final_global_chi = old_global_chi;
            final_state_weight = old_state_weight;
            final_state_seq = old_state_seq;
```

```
            final_fit_time = old_fit_time;
            final_seq_logp=final_seq_logp(1:c_fit-1);
            final_seq_global_chi=final_seq_global_chi(1:c_fit-1);
            if (nargin == 6) && (sum(abs(size(ori_param)-
size(input_param)))==0)
                re_param=re_param(1:c_fit-1,:);
            else
                re_param=[];
            end
            return
        end
        if strcmp(crit_stop,'logp')
            if new_logp > old_logp
                old_fit_param=new_fit_param;
                old_logp = new_logp;
                old_fit_tr = new_fit_tr;
                old_raw_e = new_raw_e;
                old_raw_e_fit = new_raw_e_fit;
                old_pi = new_pi;
                old_global_chi = new_global_chi;
                old_state_weight = new_state_weight;
                old_state_seq = new_state_seq;
                old_fit_time = new_fit_time;
            else
                %%% continue fitting and compare stopping criteria to
old
                %%% one

[init_tr,init_e,init_time]=extract_tre(new_fit_param,numEmis);
                c_fit=c_fit+1;
                [new2_fit_param,new2_fit_tr, new2_fit_time, new2_pi,
new2_global_chi, new2_state_seq,...
                    new2_state_weight, new2_logp, new2_raw_e,
new2_raw_e_fit, reason_stop] = ...
                    PbPHMM_unit(seq, init_tr, init_e, init_time,
input_param, new_pi);

                if isempty(new2_fit_tr)==0
                    % try
                    if nargin>=6 && (sum(abs(size(ori_param)-
size(new2_fit_param)))==0)

re_param(c_fit,:)=re_error(new2_fit_param,ori_param);
                    end
                    if (sum(sum(new2_fit_param(7:8,:)<0))~=0) &&...
                            (sum(new_fit_param(8,:))>1)
                        % p_params by both methods are wromg. Terminate
fitting.
                        % catch err
                        %     if strcmp(err.message,'Negative toff')
                        final_fit_param=old_fit_param;
                        %                 final_fit_tr = old_fit_tr;
                        final_fit_tr=old_fit_tr;
                        final_raw_e = old_raw_e;
                        final_raw_e_fit = old_raw_e_fit;
                        final_pi = old_pi;
                        final_global_chi = old_global_chi;
```

```
                        final_state_weight = old_state_weight;
                        final_state_seq = old_state_seq;
                        final_fit_time = old_fit_time;
                        final_seq_logp=final_seq_logp(1:c_fit-2);

final_seq_global_chi=final_seq_global_chi(1:c_fit-2);
                        if (nargin == 6) && (sum(abs(size(ori_param)-
size(input_param)))==0)
                            re_param=re_param(1:c_fit-1,:);
                        else
                            re_param=[];
                        end
                        return
                        %     end
                        % end
                    end
                end

                % [c_fit,toc]
                final_seq_logp(c_fit) = new2_logp;
                final_seq_global_chi(c_fit) = new2_global_chi(1);
                if
(prod(sum(new2_fit_tr,1))*prod(sum(new2_fit_tr,2))==0) || ...
                        isinf(new2_global_chi(1)) ||
(prod(sum(new2_fit_time,2))==0)...
                        || (prod(new2_state_weight)==0)
                    % Viterbi algorithm has problem at small pwt
misassigning
                    % photons at dark state as photons from bright
transitions.
                    % Therefore ignore checking zero weights.
                    %                     ||
(prod(new2_state_weight)==0)
                    % With error. finished at c_fit-2
                    final_fit_param=old_fit_param;
                    final_fit_tr=old_fit_tr;
                    final_raw_e = old_raw_e;
                    final_raw_e_fit = old_raw_e_fit;
                    final_pi = old_pi;
                    final_global_chi = old_global_chi;
                    final_state_weight = old_state_weight;
                    final_state_seq = old_state_seq;
                    final_fit_time = old_fit_time;
                    final_seq_logp=final_seq_logp(1:c_fit-2);
                    final_seq_global_chi=final_seq_global_chi(1:c_fit-
2);
                    if (nargin == 6) && (sum(abs(size(ori_param)-
size(input_param)))==0)
                        re_param=re_param(1:c_fit-1,:);
                    else
                        re_param=[];
                    end
                    return
                end
                if new2_logp > old_logp
                    old_fit_param=new2_fit_param;
                    old_logp = new2_logp;
```

```
                    old_fit_tr = new2_fit_tr;
                    old_raw_e = new2_raw_e;
                    old_raw_e_fit = new2_raw_e_fit;
                    old_pi = new2_pi;
                    old_global_chi = new2_global_chi;
                    old_state_weight = new2_state_weight;
                    old_state_seq = new2_state_seq;
                    old_fit_time = new2_fit_time;
                else
                    index = 1;
                    %            final_logp = old_logp;
                    final_fit_param=old_fit_param;
                    %                final_fit_tr = old_fit_tr;
                    %final_fit_tr_5s = old_fit_tr_5s;
                    final_fit_tr=old_fit_tr;
                    final_raw_e = old_raw_e;
                    final_raw_e_fit = old_raw_e_fit;
                    final_pi = old_pi;
                    final_global_chi = old_global_chi;
                    final_state_weight = old_state_weight;
                    final_state_seq = old_state_seq;
                    final_fit_time = old_fit_time;
                    final_seq_logp=final_seq_logp(1:c_fit-2);
                    final_seq_global_chi=final_seq_global_chi(1:c_fit-
2);
                    %                if nargin == 6 &&
(size(ori_param,2)==size(oid_fit_param,2))
                    if (nargin == 6) && (sum(abs(size(ori_param)-
size(input_param)))==0)
                        re_param=re_param(1:c_fit-2,:);
                    else
                        re_param=[];
                    end
                    return
                end
            end
        elseif strcmp(crit_stop,'chi')
            if new_global_chi < old_global_chi
                old_fit_param=new_fit_param;
                old_logp = new_logp;
                old_fit_tr = new_fit_tr;
                old_raw_e = new_raw_e;
                old_raw_e_fit = new_raw_e_fit;
                old_pi = new_pi;
                old_global_chi = new_global_chi;
                old_state_weight = new_state_weight;
                old_state_seq = new_state_seq;
                old_fit_time = new_fit_time;
            else

[init_tr,init_e,init_time]=extract_tre(old_fit_param,numEmis);
                c_fit=c_fit+1;
                [new2_fit_param,new2_fit_tr, new2_fit_time, new2_pi,
new2_global_chi, new2_state_seq,...
                    new2_state_weight, new2_logp, new2_raw_e,
new2_raw_e_fit, reason_stop] = ...
```

```
                    PbPHMM_unit(seq, init_tr, init_e, init_time,
input_param, old_pi);

                if isempty(new2_fit_tr)==0
                    % try
                    if nargin>=6 && (sum(abs(size(ori_param)-
size(new2_fit_param)))==0)

re_param(c_fit,:)=re_error(new2_fit_param,ori_param);
                    end
                    if (sum(sum(new2_fit_param(7:8,:)<0))~=0) &&...
                            (sum(new_fit_param(8,:))>1)
                        % p_params by both methods are wrong. Terminate
fitting.
                        % catch err
                        %     if strcmp(err.message,'Negative toff')
                        final_fit_param=old_fit_param;
                        %                 final_fit_tr = old_fit_tr;
                        final_fit_tr=old_fit_tr;
                        final_raw_e = old_raw_e;
                        final_raw_e_fit = old_raw_e_fit;
                        final_pi = old_pi;
                        final_global_chi = old_global_chi;
                        final_state_weight = old_state_weight;
                        final_state_seq = old_state_seq;
                        final_fit_time = old_fit_time;
                        final_seq_logp=final_seq_logp(1:c_fit-2);

final_seq_global_chi=final_seq_global_chi(1:c_fit-2);
                        if (nargin == 6) && (sum(abs(size(ori_param)-
size(input_param)))==0)
                            re_param=re_param(1:c_fit-2,:);
                        else
                            re_param=[];
                        end
                        return
                        %     end
                        % end
                    end
                end

                % [c_fit,toc]
                final_seq_logp(c_fit) = new2_logp;
                final_seq_global_chi(c_fit) = new2_global_chi(1);
                if
(prod(sum(new2_fit_tr,1))*prod(sum(new2_fit_tr,2))==0) || ...
                        (new2_global_chi(1)==Inf) ||
(prod(sum(new2_fit_time,2))==0)...
                        || (prod(new_state_weight)==0)

                    % Viterbi algorithm has problem at small pwt
misassigning
                    % photons at dark state as photons from bright
transitions.
                    % Therefore ignore checking zero weights.
```

```
                    %                       ||
(prod(new2_state_weight)==0)
                    % With error. finished at c_fit-2
                    final_fit_param=old_fit_param;
                    final_fit_tr=old_fit_tr;
                    final_raw_e = old_raw_e;
                    final_raw_e_fit = old_raw_e_fit;
                    final_pi = old_pi;
                    final_global_chi = old_global_chi;
                    final_state_weight = old_state_weight;
                    final_state_seq = old_state_seq;
                    final_fit_time = old_fit_time;
                    final_seq_logp=final_seq_logp(1:c_fit-2);
                    final_seq_global_chi=final_seq_global_chi(1:c_fit-
2);
                    if (nargin == 6) && (sum(abs(size(ori_param)-
size(input_param)))==0)
                        re_param=re_param(1:c_fit-2,:);
                    else
                        re_param=[];
                    end
                    return
                end
                if new2_global_chi < old_global_chi
                    old_fit_param=new2_fit_param;
                    old_logp = new2_logp;
                    old_fit_tr = new2_fit_tr;
                    old_raw_e = new2_raw_e;
                    old_raw_e_fit = new2_raw_e_fit;
                    old_pi = new2_pi;
                    old_global_chi = new2_global_chi;
                    old_state_weight = new2_state_weight;
                    old_state_seq = new2_state_seq;
                    old_fit_time = new2_fit_time;
                else
                    index = 1;
                    %            final_logp = old_logp;
                    final_fit_param=old_fit_param;
                    %                   final_fit_tr = old_fit_tr;
                    %final_fit_tr_5s = old_fit_tr_5s;
                    final_fit_tr=old_fit_tr;
                    final_raw_e = old_raw_e;
                    final_raw_e_fit = old_raw_e_fit;
                    final_pi = old_pi;
                    final_global_chi = old_global_chi;
                    final_state_weight = old_state_weight;
                    final_state_seq = old_state_seq;
                    final_fit_time = old_fit_time;
                    final_seq_logp=final_seq_logp(1:c_fit-2);
                    final_seq_global_chi=final_seq_global_chi(1:c_fit-
2);
                    %                   if nargin == 6 &&
(size(ori_param,2)==size(oid_fit_param,2))
                    if nargin == 6 && (sum(abs(size(ori_param)-
size(input_param)))==0)
                        re_param=re_param(1:c_fit-2,:);
                    else
```

```
                        re_param=[];
                    end
                    return
                end
            end
        else
            error('crit_stop should be either "logp" or "chi".')
        end

    end
end

if c_fit>=n_iter
    index = 1;
    %            final_logp = old_logp;
    final_fit_param=old_fit_param;
    %     final_fit_tr = old_fit_tr;
    %final_fit_tr_5s = old_fit_tr_5s;
    final_fit_tr=old_fit_tr;
    final_raw_e = old_raw_e;
    final_raw_e_fit = old_raw_e_fit;
    final_pi = old_pi;
    final_global_chi = old_global_chi;
    final_fit_time = old_fit_time;
    final_seq_logp=final_seq_logp(1:c_fit);
    final_seq_global_chi=final_seq_global_chi(1:c_fit);
    if nargin == 6 && (size(input_param,2)==size(ori_param,2))
        re_param=re_param(1:c_fit,:);
    else
        re_param=[];
    end
end
numState=length(final_fit_tr);

final_state_seq=hmmviterbi3(seq,final_fit_tr,final_raw_e',final_pi,
final_fit_time);
final_state_weight = histo_HMM(final_state_seq,numState);
final_state_weight(1) = [];
histogram=histo_HMM(seq);
histogram(1)=[];
```

```
function [fit_param,fit_tr, fit_time, fit_pi, global_chi, state_seq,
state_weight, logp,...
    raw_e, raw_e_fit] = ...
    PbPHMM_unit(obs_seq, init_tr, init_e,wait_time,
input_param,init_pi)
HMM_unit_wait_simple_pparam_newchi
%%% Return photophysical parameters and trained transition and emission
%%% matrices (corresponds to one iteration)
% fit_param : estimated photophysical parameters
% final_re_param : relative errors of estimate parameters compared to
real
%                   parameters
% fit_tr : trained transition matrix
% fit_time : expected photon waiting time of each state
% fit_pi : trained initiation matrix
% global_chi : chi-square between trained emission matrix and emission
%   matrix from trained photophysical parameters
% state_seq : reconstructed state sequence
% state_weight : population of states
% logp : log-likelihood
% raw_e : trained emission matrix
% raw_e_fit : emission matrix from trained photophysical parameters

%%% Exactly as written in Rabiner's
%%% if a fit_param(estimated photophysical parameter set) from tr and e
is
%%% not photophysically relevant, then try tr and a reconstructed state
sequence.
%%% emission probabilities of nonexisting values are set to be zero.
%%% return empty results if fit_param has an error.
%%%
% Negative scale for very small emission probabilities is corrected.
% use init_e as input. does NOT construct e from input_param.
% calculates global chi-square.
% is a unit of iteration.
% the direction of sequence in obs_seq must be horizontal.
% size(e_param) = [numState,2];
% e_param(:,1) = A, e_param(:,2) = tau
% emission matrix : exponential pdf = A*exp(-seq/tau)
%% initial photophysical parameters
% % 1. I_prim : Primary excitation intensity (W/cm2)
% % 2. e : absorption coefficient (/(M*cm))
% % 3. wavelength (nm)
% % 4. F_fl : fluorescence quantum yield
% % 5. F_det : detection efficiency
% % 6. t_fl : "measured" fluorescence lifetimea
% % 7. t_dark1 : dark state lifetime
% % 8. dark state quantum yield : k_ISC(intersystem crossing
rate)/k_rad
% % 9. ratio of k_ReISC (reverse intersystem crossing rate) to k_ISC2
% %    (triplet depopulation rate from T1 to S0) : k_ReISC/k_ISC2
% %    - Action cross section for reverse intersystem crossing due to
% %     secondary laser illumination
% % 10. background noise
% % 11. tbin : time step
% % 12. collecting time
% % 13. t_mod : modulation time (inverse of modulation frequency)
```

```
% % 14. I_secmax : Secondary excitation intensity for modulation in
W/cm^2.
% % 15. wavelength2 : wavelength of secondary laser (W/cm2)
%
t_bin = input_param(11);
% Fcol=input_param(5)*input_param(4);
% toff=input_param(7,:)/t_bin;
% Fisc=input_param(8,:);
% Fisc_sum=sum(input_param(8,:));
% I_prim=input_param(1);
% s_abs=input_param(2)*3.82356e-21;
% wavelength=input_param(3);
% t_fl=input_param(6);
% t_ic=input_param(13,1);
% back_level=input_param(10);
% I_sat = 6.6261e-34*2.9979e8/(s_abs*wavelength*10^-9*t_fl);
% %Nbins = t_mod/tbin;
% k_exc = s_abs*I_prim*wavelength*10^-9/(6.6261e-
34*2.9979e8*(1+I_prim/I_sat));
% kt=k_exc*t_bin;
fit_param=input_param;
reason_stop=[];
%% initialization
% numState = 2;
if size(obs_seq,1)<size(obs_seq,2)
    obs_seq=obs_seq';
end
numState=length(init_tr);
numEmis=max(obs_seq);
if nargin < 7
    init_pi=zeros(numState,1);
    if obs_seq(1)<3
        init_pi(2)=1;
    else init_pi(:,1) = 1/numState;
    end
end
if size(init_pi, 1) < size (init_pi,2)
    init_pi=init_pi';
end
% if size(wait_time,1) < size(wait_time,2)
%     wait_time=wait_time';
% end

L = length(obs_seq);
if sum(abs(obs_seq-round(obs_seq)))~=0
    %     hist_point = ceil(max(obs_seq)/t_bin);
    histogram = histo_HMM(ceil(obs_seq/t_bin));
    obs_seq=ceil(obs_seq/t_bin);
else
    %     hist_point = max(obs_seq);
    histogram = histo_HMM(obs_seq);
end
% hist_x=(1:hist_point)';
histogram(1)=[];

% fit_time = zeros(numState,1);
```

```
% fit_chi = zeros(numState,1);
% raw_e_fit = zeros(hist_point,numState);
%%% if min(obs_seq) == 0, length of hist = numEmis + 1
%%% if min(obs_seq) ~= 0, length of hist = numEmis


%% initial transition and emission  matrix

%%% init tr
% ds=Fcol*(1-sum(Fisc));
% dss=Fcol*(1-sum(Fisc))+sum(Fisc);
% tr with zero background
% [init_tr,init_e]=extract_tre(input_param,numEmis);

%%% make emission probabilities of nonexisting observation to be zero
init_e(histogram==0,:)=0;
for cc=1:numState
    init_e(:,cc)=init_e(:,cc)/sum(init_e(:,cc));
end
%% begin overall process%%%

%estimate forward and backward prob

%  forward variable forward_var
%  s : scaling factor

if prod(wait_time>1)==0
    % divide real wait_time by t_bin
    wait_time=wait_time/t_bin;
end



[~,logp,forward_var,backward_var,s_forward_var] =
hmmdecode_sb(obs_seq,init_tr,init_e,wait_time,init_pi);
% s_forward_var(s_forward_var<0)=exp(s_forward_var(s_forward_var<0));
%% check any NaN is in forward_var or backward_var
if
sum(isnan(forward_var(:)+backward_var(:)))+sum(isinf(abs(forward_var(:)
...
        +backward_var(:))))<3
    %% train transition matrix : Baum-Welch algorithm
    fit_tr=zeros(numState,numState,L-1);
    for current = 1:numState
        for post = 1:numState
            %             fit_tr(k,l,:)=forward_var(1:L-
1,k)*init_tr(k,l).*init_e(obs_seq(2:L),l).*...
            %                 backward_var(2:L,l)./s_forward_var(1:L-
1);
            fit_tr(current,post,:)=forward_var(1:L-
1,current)*init_tr(current,post).*...
                init_e(obs_seq(2:L),post).*backward_var(2:L,post);
        end
    end
    % by Rabiner's
    %     gamma=exp(log(forward_var)+log(backward_var));
```

```
    %     gamma=gamma./repmat(sum(gamma,2),1,numState);
    %     gamma=sum(gamma,1);
    %     for current = 1:numState
    %         for post = 1:numState
    %             fit_tr(current,post,:)=forward_var(1:L-
1,current)*init_tr(current,post).*...
    %
init_e(obs_seq(2:L),post).*backward_var(2:L,post)/gamma(current);
    %         end
    %     end


    % remove nan (replace by zero)
    fit_tr(isnan(fit_tr))=0;
    fit_tr(isinf(fit_tr))=0;
    fit_tr(fit_tr<0)=0;
    fit_tr=norm_m(sum(fit_tr,3));
    %check if all states are accessible
    if prod(sum(fit_tr,1))==0 || prod(sum(fit_tr,2))==0
        reason_stop = [reason_stop,2];
        fit_tr=[];
        fit_time=[];
        fit_pi=[];
        global_chi=inf;
        state_seq=[];
        state_weight=[];
        logp=-inf;
        logp_raw=-inf;
        raw_e=[];
        raw_e_fit=[];
        sum_e=[];
        sum_e_fit=[];
        return
    end
    %% train emission matrix
    % for numerical stability, put logarithm
    gamma=exp(log(forward_var)+log(backward_var));
    gamma=gamma./repmat(sum(gamma,2),1,numState);
    %replace NaN in gamma by zero
    gamma(isnan(gamma))=0;
    raw_e=zeros(size(init_e));
    for count = 1:L
        for state = 1:numState
            % mine
            raw_e(obs_seq(count),state) =
raw_e(obs_seq(count),state)...
                + gamma(count,state);
            % Rabiner's : gamma(state)/sum(gamma(state)), but it's
            % normalized already.
            %old : + gamma(count+1,state);
        end
    end
    %normalize the trained emission matrix
    raw_e = raw_e./repmat(sum(raw_e,1),size(raw_e,1),1);
    fit_pi = gamma(1,:);
```

```
    %     clear forward_var backward_var gamma s_forward_var init_tr
init_e
    %% check state occupation. if any state is not occupied, terminate
and return void
    fit_time=zeros(numState,1);
    for cc=1:numState
        fit_time(cc)=sum((1:numEmis)'.*raw_e(:,cc))*t_bin;
    end
%     state_seq=hmmviterbi3(obs_seq,fit_tr,raw_e_fit,fit_pi,fit_time);
    state_seq=hmmviterbi3(obs_seq,fit_tr,raw_e,fit_pi,fit_time);
    %% extract p-param

    % state probability or pwts by Viterbi algorithm has larger
    % error than Baum-Welch algorithm. So calculate pwt by tr and e
    % rather than assigned states. index_pwt --> 0

fit_param=extract_p_param_num(obs_seq,state_seq,fit_tr,raw_e,input_para
m,0);

    if prod(prod(fit_param(7:8,:)))==0
        % if fit_param from tr and e returns error, try a tr and the
state
        % sequence

fit_param=extract_p_param_num(obs_seq,state_seq,fit_tr,raw_e,input_para
m,1);
        if prod(prod(fit_param(7:8,:)))==0
            % if the fit_param still has an error, return error
results.
            reason_stop=[reason_stop,3];
%             fit_param=[];
            fit_tr=[];
            fit_time=[];
            fit_pi=[];
            global_chi=Inf;
            state_seq=[];
            state_weight=[];
            logp=-inf;
            logp_raw=-inf;
            raw_e=[];
            raw_e_fit=[];
            return
        end
    end

    %% Determine raw_e_fit ==> extracted from fitted p_param
    % need no more curve fitting
    % fit_time : mean pwt of each state.
    [~,raw_e_fit,fit_time]=extract_tre(fit_param,numEmis);
    try
        state_weight = histo_HMM(state_seq, numState);
        state_weight(1) = [];
%         sum_e = sum( raw_e.* repmat(state_weight',numEmis,1),2);
        sum_e_fit = sum( raw_e_fit.*
repmat(state_weight',numEmis,1),2);
```

```
        global_chi = est_real_chi(histogram, sum_e_fit, numEmis-
numState);
        % calculate logp again with trained tr and e_fit.
        [~,logp] =
hmmdecode_sb(obs_seq,fit_tr,raw_e_fit,sum(fit_time,2),fit_pi);
        [~,logp_raw] =
hmmdecode_sb(obs_seq,fit_tr,raw_e,sum(fit_time,2),fit_pi);
    catch err
        % if parameter extraction does not work
        if isempty(err.identifier)==0
            fit_param=[];
            fit_tr=[];
            fit_time=[];
            fit_pi=[];
            global_chi=Inf;
            state_seq=[];
            state_weight=[];
            logp=-inf;
            logp_raw=-inf;
            raw_e=[];
            raw_e_fit=[];
            sum_e=[];
            sum_e_fit=[];
        end
    end

else
    fit_param=[];
    fit_tr=[];
    fit_time=[];
    fit_pi=[];
    global_chi=Inf;
    state_seq=[];
    state_weight=[];
    logp=-inf;
    logp_raw=-inf;
    raw_e=[];
    raw_e_fit=[];
    sum_e=[];
    sum_e_fit=[];
end


function [val_chi, val_reduced_chi] = est_real_chi(ori_seq, fit_seq,
df)
if size(ori_seq,2) > size(ori_seq,1)
    ori_seq = ori_seq';
end
if size(fit_seq,2) > size(fit_seq,1)
    fit_seq = fit_seq';
end
if size(ori_seq,2) ~= size(fit_seq,2)
    error('The size of two sequences must be the same.')
end
%numStep = size(ori_seq,2);
temp_chi = (ori_seq - fit_seq).^2./fit_seq;
```

```
sum_inf = sum(isinf(temp_chi),1);
sum_nan = sum(isnan(temp_chi),1);
if sum_inf + sum_nan == size(ori_seq,2)
    val_reduced_chi = inf;
    val_chi = inf;
else
    temp_chi(isnan(temp_chi)) = 0;
    temp_chi(isinf(temp_chi)) = 0;
    val_chi = sum(temp_chi);
    val_reduced_chi = val_chi/df;
end

function [histogram, x_hist] = histo_HMM(seq, bin)
%%% calculate histogram of seq
%%% bin = 1
%%% length of result = max of seq + 1
%%% x_hist : x axis of histogram ( 0 <= x_hist <= max(seq)
if nargin<2
    bin=max(seq);
end
numStep1 = size(seq,1);
numStep2 = size(seq,2);
x_hist = 0:max(max(seq));
histogram = zeros(bin+1,1);
for count1 = 1:numStep1
    for count2 = 1:numStep2
        if seq(count1,count2) == 0
            histogram(1) = histogram(1) + 1;
        else
            histogram(seq(count1,count2)+1) =
histogram(seq(count1,count2)+1) + 1;
        end
    end
end

function [m1,m2,fit_seq,rsq]=fit_conv_twoexp(x,y)
%f(x)=1/m1*exp(-x/m1)
%f(y)=1/m2*exp(-y/m2)
if size(x,1)<size(x,2)
    x=x';
end
if size(y,1)<size(y,2)
    y=y';
end
y2=y/sum(y.*(x-[0;x(1:end-1)]));
init_m=sum(x.*y2.*(x-[0;x(1:end-1)]));
fit_sum = ezfit(x,y2,['y=1/(m2-m1)*(exp(-x/m2)-exp(-x/m1));m1=',...
    num2str(init_m/2),';m2=',num2str(init_m/2)]);
fit_sum = ezfit(x,y2,['y=1/(m2-m1)*(exp(-x/m2)-exp(-x/m1));m1=',...
    num2str(fit_sum.m(1)),';m2=',num2str(fit_sum.m(2))]);
% a=t_on, b=t_off
m1=fit_sum.m(1);
m2=fit_sum.m(2);
rsq=fit_sum.r;
fit_seq=1/(m2-m1)*(exp(-x/m2)-exp(-x/m1));
```

```
function [seq_rate, fit_seq, val_chi] = fit_erlang(seq, x, k)
if size(seq,1) < size(seq,2)
    seq=seq';
end
if size(x,1) < size(x,2)
    x=x';
end
L = size(seq,1);
init_l = sum(x.*(x-[0;x(1:end-1)]).*seq)/k;
i_step = fix(log10(init_l))-1;
l_seq = zeros(20,2);
%%calculate and compare squared sum of residuals(integer).
for c1=1:10
    for count = 1:20
        if init_l + 10^(i_step-c1+1)*(count-10) > 0
            l_seq(count,1) = init_l + 10^(i_step-c1+1)*(count-10);
            dd = seq - erlang(x, k, l_seq(count,1));
            l_seq(count,2) = sum(dd.^2/(L-1),1);
        else l_seq(count,2) = Inf;
        end
    end
    init_l = l_seq(argmin(l_seq(:,2)),1);
end
seq_rate = init_l;
val_chi = min(l_seq(:,2));
fit_seq = erlang(x,k,seq_rate);
clear l_seq l_seq2 l_seq3 l_seq4


function indices = argmin(v)
% ARGMIN Return as a subscript vector the location of the smallest
element of a multidimensional array v.
% indices = argmin(v)
%
% Returns the first minimum in the case of ties.
% Example:
% X = [2 8 4; 7 3 9];
% argmin(X) = [1 1], i.e., row 1 column 1

[m i] = min(v(:));
indices = ind2subv(mysize(v), i);


function indices = argmax(v)
% ARGMAX Return as a subscript vector the location of the largest
element of a multidimensional array v.
% indices = argmax(v)
%
% Returns the first maximum in the case of ties.
% Example:
% X = [2 8 4; 7 3 9];
% argmax(X) = [2 3], i.e., row 2 column 3

[m i] = max(v(:));
```

```
indices = ind2subv(mysize(v), i);

function sz = mysize(M)
% MYSIZE Like the built-in size, except it returns n if M is a vector
of length n, and 1 if M is a scalar.
% sz = mysize(M)
%
% The behavior is best explained by examples
% - M = rand(1,1),   mysize(M) = 1,      size(M) = [1 1]
% - M = rand(2,1),   mysize(M) = 2,      size(M) = [2 1]
% - M = rand(1,2),   mysize(M) = 2,      size(M) = [1 2]
% - M = rand(2,2,1), mysize(M) = [2 2],  size(M) = [2 2]
% - M = rand(1,2,1), mysize(M) = 2,      size(M) = [1 2]

if isvector(M)
    sz = length(M);
else
    sz = size(M);
end
```

```
function re_param=re_error(fit_param,ori_param)
%%% calculate relative error of fit_param with ori_param

%%% old re
%%% re_param(c_fit,:)=[(new2_fit_param(7,:)-
ori_param(7,:))./ori_param(7,:),...
%%%     (new2_fit_param(8,:)-ori_param(8,:))./ori_param(8,:)]*100;
%%% re_param(c_fit,:)=log([new2_fit_param(7,:),new2_fit_param(8,:)]...
%%%     ./fp_ori)/log(2)*100;
num_param=size(fit_param,2);
if sum(abs(size(fit_param)-size(ori_param)))~=0
    re_param=0;
else
    fit_param=[fit_param(7,:),fit_param(8,:)];
    ori_param=[ori_param(7,:),ori_param(8,:)];
    re_param=zeros(1,num_param*2);
    for cc=1:num_param*2
%         re_param(cc)=(fit_param(cc)-ori_param(cc))/...
%             min([fit_param(cc),ori_param(cc)])*100;
        re_param(cc)=(fit_param(cc)-ori_param(cc))/ori_param(cc)*100;
    end
end
```

```
function raw_data = SPC_read(spc_file, setup_file)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% convert SPC file from TCSPC module into the following format
% The file name of setup and spc files must be between apostrophes.
%
% Example : raw_data = SPC_read('exp1.spc', 'exp_set.set');
%
% setup_file : the name of setup file. If not typped, the code will
find a
% setup file with the same name of spc_file.
% Output data(raw_data) have 5 columns. Columns have following
parameters.%
% 1. Absolute Macro Time clocks  2. Micro time (in sec)
%
% 3. Real Macro Time clocks (including micro)  4. photon waiting time
%
% 5. Channel 6.Invalid flag
%
%
%
%                             - 09/15/2010 Soonkyo Jung
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
if nargin < 2
    setup_file = [spc_file(1:end-3),'set'];
end
fid1=fopen(setup_file,'r');
raw_setup = fread(fid1);
fclose(fid1);
fid2=fopen(spc_file,'r');
spc_data = fread(fid2);
fclose(fid2);
br_list=find(raw_setup==93);
po1=br_list(23);
po2=br_list(24);
po3=br_list(25);
TACrange=str2double(char(raw_setup(po1+21:po2-1)));
TACgain=str2double(char(raw_setup(po2+21:po3-1)));
% convert spc data into 6 columns raw data
% column name

% size_raw due to the fact that the information per photon is stored as
6 bytes.
size_raw = length(spc_data)/6-1;
raw_data = zeros(size_raw, 6);
% mt_base : when the macro timer overflows 2^24, previous macro
% timer is added to this value.
mt_base = 0;
clock = (spc_data(3)+256*spc_data(4))*1e-10;
for count = 1:size_raw
    if spc_data(6*count+2) < 16
        adc = spc_data(6*count+2)* 256 + spc_data(count*6+1);
        raw_data(count,6) = 0;
        mtov = 0;
    elseif spc_data(6*count+2) < 32
```

```
        adc = (spc_data(6*count+2)-16)* 256 + spc_data(count*6+1);
        raw_data(count,6) = 1;
        mtov = 0;
    elseif spc_data(6*count+2) < 48
        adc = (spc_data(6*count+2)-32)* 256 + spc_data(count*6+1);
        raw_data(count,6) = 0;
        mtov = 1;
    elseif spc_data(6*count+2) < 64
        adc = (spc_data(6*count+2)-48)* 256 + spc_data(count*6+1);
        raw_data(count,6) = 1;
        mtov = 1;
    end
    if mtov == 1
        mt_base = mt_base + 16777216;
    end
    raw_data(count,2) = (4095 - adc) * TACrange/TACgain/4096;
    raw_data(count,1) = spc_data(6*count+3)*65536 +
spc_data(6*count+5)...
        + spc_data(6*count+6) * 256 + mt_base;
    raw_data(count,3) = raw_data(count,1) * clock + raw_data(count,2);
    raw_data(count,5) = spc_data(6*count+4);
end
raw_data(:,4)=raw_data(:,3)-[0;raw_data(1:end-1,3)];
%raw_data(:,3)=cumsum(raw_data(:,3));
```